ABSTRACT

Title of thesis: MULTI-AGENT UUV VALIDATION VIA ROLLING-HORIZON ROBUST GAMES

Kevin J. Quigley, Master of Science, 2019

Thesis directed by: Professor Steven Gabriel
Department of Mechanical Engineering

Autonomy in unmanned underwater vehicle (UUV) navigation is critical for most applications due to inability of human operators to control, monitor or intervene in underwater environments. To ensure safe autonomous navigation, verification and validation (V&V) procedures are needed for various applications. This thesis proposes a game theory-based benchmark validation technique for trajectory optimization for non-cooperative UUVs. A quadratically constrained nonlinear program formulation is presented, and a "perfect-information reality" validation framework is derived by finding a Nash equilibrium to various two-player pursuit-evasion games (PEG). A Karush-Kuhn-Tucker (KKT) point to such a game represents a best-case local optimum, given perfect information available to non-cooperative agents. Rolling-horizon foresight with robust obstacles are incorporated to demonstrate incomplete information and stochastic environmental conditions. A MATLAB-GAMS interface is developed to model the rolling-horizon game, and is solved via a mixed complementarity problem (MCP), and illustrative examples show how equilibrium trajectories can serve as benchmarks for more practical real-time path planners.

MULTI-AGENT UNMANNED UNDERWATER VEHICLE VALIDATION VIA ROLLING-HORIZON ROBUST GAMES

by

Kevin J. Quigley

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2019

Advisory Committee:
Professor Steven A. Gabriel, Chair/Advisor
Professor Shapour Azarm
Professor Michael Fu
Professor Jeffrey W. Hermmann

Acknowledgments

This thesis would not have been possible without the contributions, encouragement and insight from my professors and colleagues during my studies at the University of Maryland.

To my advisor, Dr. Steven Gabriel, thank you for your consistent guidance, wisdom and patience throughout the course of this research, and for giving me the opportunity to learn and grow through this experience.

Thanks to the other members of the Lockheed Martin V&V challenge project team, on which I was a research participant (grant MRA17001004), Dr. Shapour Azarm, Tianchen Liu and Xiangxue Zhao, whose collaboration and recommendations were crucial to my research.

Thanks also to my committee members Dr. Michael Fu and Dr. Jeffrey Herrmann for their invaluable advice to improve the language and clarity of this document.

A special thanks goes to my wife Lauren, daughter Kenzie and son James for their unwavering love, support and sacrifices.

Lastly, thank you to God for providing the opportunities, means and motivation to live a life of learning.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviations

GAMS	General Algebraic Modeling System software
KKT	Karush-Kuhn-Tucker
MATLAB	Matrix Laboratory software
MCP	mixed complementarity problem
NLP	nonlinear program
PEG	pursuit-evasion game
UUV	unmanned underwater vehicle
V&V	verification and validation

Nomenclature

a_{max}	maximum acceleration
A	linear dynamics state matrix
b	linear dynamics initial condition vector
B	linear dynamics control matrix
c_m	center of obstacle m
d_h	displacement (distance traveled) in horizon h
f_G, f_P, f_E	goal, pursuit, energy sub-objective functions
$J_h^{[i]}$	cost functional for vehicle i in horizon h
$p_{k,d}^{[i]}$	position of vehicle i in dimension d at time step k
r_h	travel range in horizon h
r_m	radius of obstacle m
r_δ	detection radius
$s_h^{[i]}$	strategy of vehicle i in horizon h
T_h	horizon length
Δt	time step duration
$u_k^{[i]}$	control (acceleration) of vehicle i at time step k
$v_{k,d}^{[i]}$	velocity of vehicle i in dimension d at time step k
v_{max}	maximum velocity
$x_k^{[i]}$	state (position and velocity) of vehicle i at time step k
$\mu, \sigma, \nu, \lambda$	Lagrangian multipliers
$\omega_G, \omega_P, \omega_E$	multi-objective weighting parameters
γ, δ	detection indicators
$\varepsilon_G, \varepsilon_C$	termination tolerances

Chapter 1

Introduction

1.1 Motivation

The expansion of new technologies in autonomous systems, particularly in unmanned vehicles, provides opportunities for innovative improvements across industries, principally to improve upon time and energy efficiency, safety, and cost-effectiveness of dangerous or monotonous tasks. With applications in oceanography, deep sea exploration, offshore commercial infrastructure monitoring and maritime defense, unmanned underwater vehicles (UUVs) have a wide range of mission sets, characteristics and requirements [1]. Each mission set inherently requires advances in autonomy, energy and propulsion, sensors and sensor processing, communications/navigation, and engagement/intervention [10]. While these components all significantly impact the performance of UUVs, the research area of autonomy and control is critically important due to its operating environment and presents the greatest challenge. In such a complex and often restricted environment, UUVs require significantly more robust autonomy than do other unmanned systems due to limited–and often lack of--communication with human controllers [10].

As the roles and capabilities of unmanned systems expand to improve autonomous performance, the test, evaluation, verification and validation (TEVV) becomes a critical factor in building confidence in autonomy. For adaptive and non-

deterministic systems, a new approach to TEVV is needed, especially considering emergent behavior from interacting autonomous systems [9].

The rapid propagation of unmanned systems into various applications across industries will presumably continue, and there is little doubt that unmanned underwater vehicles will be at the forefront of maritime exploration, ocean engineering and naval defense. Due to the inherent difficulty in maintaining a communication link between human controllers and underwater vehicles, the autonomy required for UUVs needs to be significantly more sophisticated than in other unmanned systems [10]. The need for reliable autonomy implies a demand for the verification and validation (V&V) of UUVs across various mission sets and vehicle configurations.

1.2 Problem Description

This thesis outlines a game theory-based approach to validation for current and future path-planning methods for unmanned underwater vehicles. There exist many methods developed and improved upon over the past two decades in the realm of path planning for autonomous systems, primarily in determining optimal paths and ensuring collision avoidance for wheeled and aerial vehicles. Generally, the behavior of these systems can be described as a function of time-varying decision variables, state variables, and uncertain parameters.

For the purposes of this study, the term "agent" refers to a single UUV or a team thereof, with unique objective(s) and constraints. Figure 1.1 describes the spatio-temporal system considered in this thesis. The state variables $x(t)$ refers

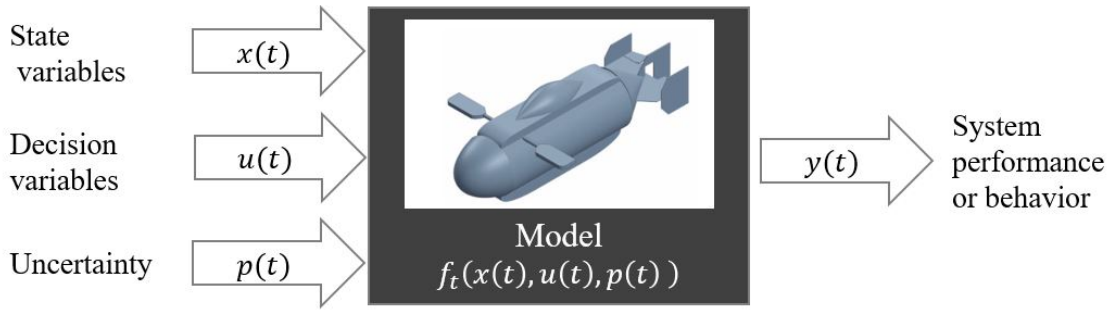


Figure 1.1: Spatiotemporal System with General Variables [23]

to the current vehicle position and velocity at each time step, while the decision variables $u(t)$ refers to thrust (in the form of acceleration) along an intended path. Uncertain parameters $p(t)$ refer to the external (exogenous) uncertain conditions in the environment such as size and position of obstacles, water currents, etc. Uncertainty can also arise from internal (endogenous) conditions due to learning. The objectives of each agent (represented by performance $y(t)$) differ based on their role.

1.3 Focus of this Research

The goal of this research is to develop a scenario-driven benchmarking technique to validate current and future UUV trajectory-planning models as they interact with other agents via rolling-horizon robust non-cooperative games. A particular two-player, pursuit-evasion game is presented for two rigid-body robotic systems controlled by adversarial agents denoted as the "attacker" and "defender," respectively. Figure 1.2 provides a simple visual example of the pursuit-evasion game. The red diamond represents the attacker UUV, which selects a path (red dashed line) based on a multiple objectives to reach a target point (green triangle) while

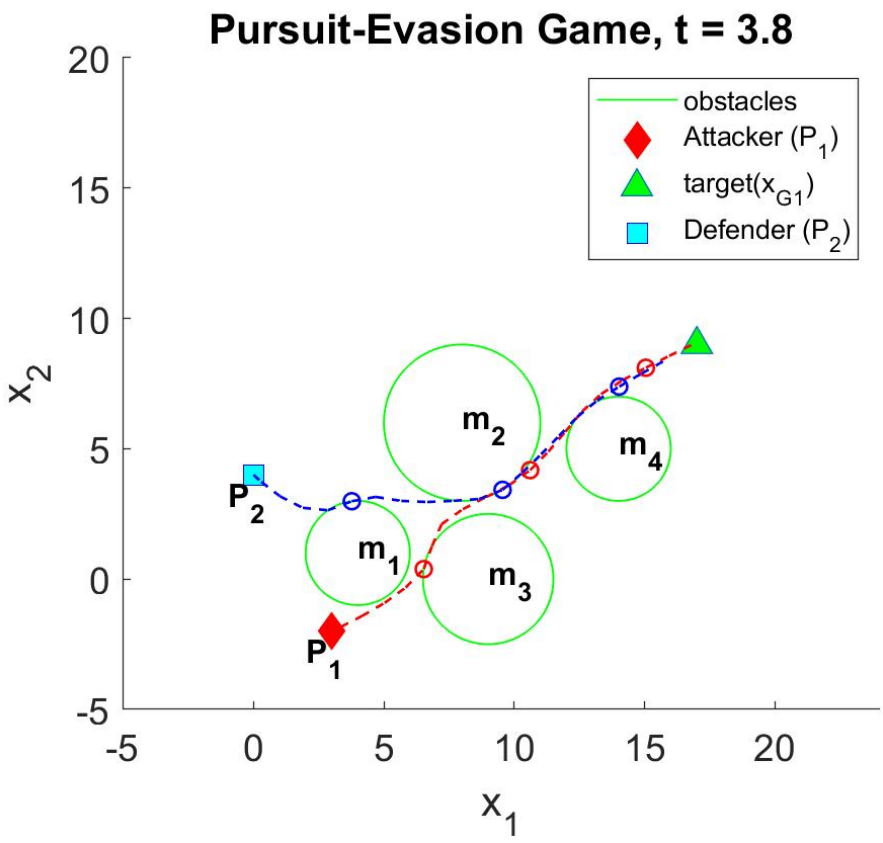


Figure 1.2: Two-player Pursuit-Evasion Game

maximizing the distance from the defender UUV. The defender, represented by the blue square, has an objective to intercept the attacker, and selects a path (blue dashed line) that minimizes the distance between the two. These objectives, along with common constraints will be described in depth in the following chapters.

This thesis combines the concepts of mixed complementarity problems (MCP), rolling-horizon foresight with incomplete information, stochasticity, and an infinite state space. While the majority of this thesis considers the specific application of military UUVs, the formulations presented can be generally applied across vehicle platforms and dimensions. For instance, the model presented could also be modified to reproduce wildlife tracking and monitoring scenarios, open water search and

rescue, drug-traffic interdiction, or even life guarding.

1.4 Literature Review

This section identifies some of the methods and considerations in the current literature, which help to set the conditions for the research upon which this thesis is based.

1.4.1 Path-Planning Methods for Unmanned Systems

While validation of unmanned systems can refer to the control systems, structural design, material selection, or human-machine interface, the focus of this thesis is path and trajectory planning. Chapter 2 presents a constrained nonlinear program that models the trajectory for a single UUV, with non-convex obstacle avoidance, whose quadratic objective it is to minimize the distance to a stationary target.

This thesis discretizes time into constant time steps and models motion using *direct collocation* as defined in [20]. Continuous time, position, velocity and acceleration are mapped onto discrete vectors associated with times $t_0, t_1, ..., t_K$ called *collocation points*. In trajectory planning, there is typically some continuous integral to be minimized in the objective function, which can be approximated via a summation, made possible by these collocation points.

Even with discrete time steps, obstacles along a potential trajectory in an infinite state-space present challenges in the form of non-convex feasibility constraints. Current trajectory optimization methods in continuous space generally use

a sequential convexification technique to address non-convex obstacle constraints. These include [45], which describes an algorithm called $\mathcal{A}_{scp}$ for unmanned aerial vehicles that leverages the existing A^* (a best-first search heuristic based on Dijkstra's algorithm) to compute an initial guess for its sequential convex programming model. The authors find that $\mathcal{A}_{scp}$ globally converges to local optimum (specifically, a Karush-Kuhn-Tucker point) at each time step, which is of particular interest for finding an equilibrium point in path planning for multiple agents with opposing objectives. However, $\mathcal{A}_{scp}$ requires a mixed-integer program to be initialized, which is not conducive to finding equilibrium solutions. A similar approach, SCvx [28], [25] and [12] uses lossless successive convexification and trust regions to converge to optimal trajectories. To improve computation time of finding sequential solutions, [8] presents the decoupled multi-agent iSCP technique. Other methods, namely the optimization-based collision avoidance (OBCA) scheme [44], leverage second-order cone programming and strong duality to enforce the non-convex obstacle avoidance constraints.

While the non-convexity of obstacle avoidance constraints presents limitations, the focus of this thesis is not to achieve guaranteed optimality, but to validate multi-agent feasibility of UUV trajectory-planning methods when interaction among non-cooperative agents influences decisions. Thus, the benchmark model retains non-convexity, but demonstrates a good success rate using even elementary initialization.

1.4.2 Game Theory in Path-Planning

There exist both cooperative and non-cooperative game theoretic representations of unmanned system motion planning. For instance, the time-optimal multi-stage open-loop game formulation presented in [39] offers a variant of the fast marching method for shortest path problems to find safe-reachable paths in a pursuit-evasion differential game. On the other hand, [18] leverages Voronoi partitioning to generate a guaranteed capture decentralized control scheme for a multiple pursuer-single evader game in the plane. Specific to UUVs, [13] presents a neural network-based approach to finding an equilibrium solution via the Minimax algorithm to an asymmetric skirmish between an autonomous underwater vehicle (AUV) and a manned submarine. Meanwhile, [41] presents an open-loop Defender-Attacker-Target Game and [24] describes a distributed cooperative game-theoretic control algorithm for area coverage.

A brief overview of game theoretic concepts is presented in Chapter 2, while Chapter 3 presents a solution to the non-linear program via Karush-Kuhn-Tucker optimality conditions and presents a small multi-player example. In the pursuit-evasion game (PEG) scenario described in Chapter 3, the "attacker" agent's objective is to most efficiently reach its target destination while avoiding interception by other vehicle(s). Meanwhile, the "defender" agent's mission is to minimize the distance to the nearest attacker vehicle.

This scenario of autonomous unmanned vessels swimming around independently, identifying and engaging enemy assets is of particular interest to naval lead-

ers in studying anti-submarine warfare [1]. This problem can also be translated to the "time critical strike" (TCS) naval mission set, though UUVs seem best suited for delivering weapons caches rather than autonomous weapon launches due to international man-in-the-loop launch control requirements [14].

1.4.3 Uncertainty and Foresight in Underwater Navigation

The nature of underwater systems sets them apart from other unmanned systems due primarily to the difficulties involved with human-machine interaction, as well as other environmental complexities. Unlike unmanned aerial vehicles (UAVs) or unmanned ground vehicles (UGVs), line-of-sight quickly becomes an operating constraint on manual control and/or monitoring [34]. Due to inconsistent GPS reliability underwater, UUV navigation is particularly challenging, and depends on dead reckoning (i.e., setting a heading from a known point to a destination and traveling along a constant straight trajectory), inertial and gyroscopic sensors [34]. Additionally, sonar performance is a complex function of transmission loss, reverberation and noise levels [13]. These parameters can vary based on water salinity, temperature, surface conditions, depth of the vehicle, and sea-bottom shape [13]. For these reasons, detection of obstacles and other vehicles can become difficult at any range and highly unreliable at long ranges. Thus, in Chapter 4, rolling-horizon foresight is presented in order to simulate periodic sonar updating and limited field of view that exists underwater. Rolling-horizon foresight is not, in itself, a new concept. Several current models [37], [17] consider similar receding horizons for

mixed-integer program path planners for multiple vehicles, while [38] uses receding horizons for online tube model predictive control (TMPC). Rolling horizons are also often used in approximate dynamic programming (ADP) [33], which optimizes near-term decisions for large state spaces while approximating the value of future decisions.

Uncertainty can arise in many forms. Uncertainty in the vehicle dynamics can be represented implicitly through symbolic bounds on velocity and acceleration [31], while environmental uncertainty is often represented in the form of obstacles with stochastic location, size or velocity. Much of the literature accounts for obstacle avoidance by modeling a safe path via a "tube" around a nominal trajectory [27], [38], or by inflating the obstacles to provide a buffer region [44], [32]. The model presented attempts to find collision free trajectories with static obstacles, using the latter approach to admit some uncertainty in the obstacles it perceives within a specified interval. Thus, robust obstacles are also presented in Chapter 4, which simulate the inherent error that propagates in water over long ranges, even utilizing forward-looking sonar and video for obstacle avoidance.

1.4.4 Research Contributions

The concepts critical to this research are partly and independently examined in the current literature, as described in Table 1.1. Path-planning and trajectory-optimization have been investigated extensively in the literature, though as of yet, none have been discovered that combine stochastic environmental conditions (ro-

Table 1.1: Research Contributions

Research Areas	∗	Existing Literature									
		[30]	[20]	[31]	[27]	[38]	[37]	[45]	[17]	[40]	[41]
Unmanned System V & V	✓	✓		✓							
UUV Path Planning	✓		✓						✓		
Continuous-Space Trajectory	✓		✓	✓	✓	✓					
KKT Optimality	✓				✓			✓			
Multi-Agent Model	✓						✓		✓	✓	✓
Non-cooperative Games	✓									✓	✓
Nash Equilibrium	✓								✓		
Rolling-Horizon Foresight	✓					✓	✓		✓		
Stochastic Obstacle Avoidance	✓					✓				✓	

bust obstacles), rolling-horizon foresight (incomplete information) and adversarial influences (non-cooperative games).

This thesis combines all these components in order to establish a game theory-based trajectory benchmark tool. This tool can be implemented to check the functional feasibility of practical path-planning tools using common parameters.

The model presented seeks to establish a multi-player, rolling-horizon, robust equilibrium. For the purposes of validation, a solution to such a model can serve as a "perfect-information reality" benchmark. Benchmarking in this sense provides a lower bound best-case feasible trajectory strategy for each UUV, assuming all other agents have perfect information of its action within each planning horizon. This can be used to validate the feasibility of solutions from non-learning, independent trajectory optimization models, specifically in non-cooperative scenarios.

Specifically, a simple two-player pursuit-evasion game (PEG) is built using a General Algebraic Modeling System (GAMS)-MATLAB interface [6], [29], [11]. This simulation is run using a constrained nonlinear program formulation in various two-dimensional environmental configurations to determine their respective accu-

racy with respect to a Karush-Kuhn-Tucker (KKT), or equilibrium, point at each time step. The results of this simulation and conclusions drawn are presented. Future work could generalize the approach to more complex multi-agent systems and incorporate vehicle-based, rather than geocentric dynamics. In addition, the non-convex obstacle constraint qualifications could be investigated analytically, or improved upon by incorporating convexification techniques and reducing inevitable collision states.

Chapter 2

Non-Convex Optimal Trajectory Problem

2.1 Problem Description

To solve a continuous-time, optimal control problem via optimization methods, we should first discretize the trajectory [28], giving us a finite set of decision variables. To do so, we use direct collocation as in [20], mapping continuous position $p(t)$, velocity $v(t)$ and acceleration $u(t)$ (control) to their values at specific instances, known as collocation points:

$$t \to t_0, ..., t_k, ..., t_N, \quad x(t) \to x_0, ..., x_k, ..., x_N, \quad u(t) \to u_0, ..., u_k, ..., u_N.$$

Note that the notation $\to$ means "maps to" in this case. Though the problem can be extended to three dimensions for practical use, we consider only two dimesions ($n = 2$) in this thesis. Thus, the state vector x_k represents both the position $p_k \in \mathbb{R}^2$ and velocity $v_k \in \mathbb{R}^2$ at the k^{th} discrete time step, such that $x_k = (p_k^T, v_k^T)^T$. The control vector $u_k \in \mathbb{R}^2$ similarly represents vehicle acceleration at the k^{th} time step.

So then

$$x_k = \begin{bmatrix} p_{k,1} \\ p_{k,2} \\ v_{k,1} \\ v_{k,2} \end{bmatrix}, \quad u_k = \begin{bmatrix} u_{k,1} \\ u_{k,2} \end{bmatrix},$$

where $p_{k,1}$ and $p_{k,2}$ are the position in the horizontal and vertical dimensions, respectively. Also, similar notation holds for the velocity v_k and acceleration u_k. Without loss of generality, we can assume N, the last time period, is a fixed integer. Thus, along the entire horizon $T_h = t_N - t_0$, we can summarize these variables

$$x \triangleq (x_0^T, ..., x_k^T, ..., x_N^T)^T \in \mathcal{X} \subseteq \mathbb{R}^{4(N+1)}$$

$$u \triangleq (u_0^T, ..., u_k^T, ..., u_{N-1}^T)^T \in \mathcal{U} \subseteq \mathbb{R}^{2N}$$

where $\mathcal{X}$ is the Cartesian product of $\mathcal{X}_k$, and $\mathcal{U}$ is the Cartesian product of $\mathcal{U}_k$. This notation for the state and control vectors, (x, u), will also later be extended to a rolling-horizon in Chapter 4, where a sub-problem state vector $x_{(h)}$ and control vector $u_{(h)}$ are found in each planning horizon h. Note that the size of the state vector x contains both the position and velocity for all time steps, so is slightly more than twice the size of the control vector u, which contains only the acceleration terms at each time step, except the final step k_N. The state and control domains, $\mathcal{X}_k$ and $\mathcal{U}_k$, are assumed to be non-empty, convex and compact sets which include boundary

conditions [28]. For example, the initial conditions give $\mathcal{X}_0 = x_{IC}$, where x_{IC} is the known starting position and velocity. The maximum acceleration (due to thrust, drag, inherently due to thrust capacity, drag, bouyancy, etc.) is captured in $\mathcal{U}_k$. It is sufficient then to consider the discrete-time, finite-horizon, optimal control problem, which we presently define for a specific case.

2.2 Two-Dimensional Optimal Trajectory for a Single UUV

2.2.1 Goal Function

Consider the simple path-planning problem for a single "attacker" vehicle represented as a point mass in two dimensions ($n = 2$), seeking an optimal trajectory to a known target position $p_G(k)$ in an environment with obstacles that restrict movement. Let $\mathcal{W}$ denote the workspace of our system, which contains all possible vehicle configurations, $\mathcal{X} \subseteq \mathcal{W}$, in addition to a set of obstacles $\mathcal{O} \subseteq \mathcal{W}$, whose locations and dimensions are subject to uncertainty. In Chapter 4, the obstacle size and locations can vary from the expected configuration, which will necessitate finding a robust feasible region.

Assuming that a path exists, our goal is to determine a state and control sequence (x, u) that ensures safe navigation from the initial state x_{IC} to its target state x_G, while minimizing a cost functional $J(x, u) = \sum_{k=0}^{N} f_G(x_k, u_k)$, (2.1a), subject to initial conditions x_{IC} (2.1b), linear dynamics (2.1c), state and control bounds (2.1d)-(2.1e), and avoiding collision with N_{obs} obstacles $\mathcal{O}_{(1)}, ..., \mathcal{O}_{(m)}, ..., \mathcal{O}_{(N_{obs})} \in \mathcal{O}$. We

define this cost function at the k^{th} time step as

$$f_G(x_k, u_k) = \frac{1}{2}\|p_G(k) - p_k\|^2, \tag{2.1a}$$

in which $\|\cdot\|$ is the L_2-norm, and thus convex and continuously differentiable in the decision variables. We call f_G the "goal function"— the straight line distance between the vehicle's target $p_G(k)$ and its position p_k.

2.2.2 Linear Dynamics

A feasible path needs to satisfy physical dynamics, as well as bounds on velocity, $v_{max}(k)$, and acceleration, $a_{max}(k)$, due to a combination of vehicle-platform characteristics and environmental conditions. Though vehicle dynamics are typically nonlinear, they can be linearized via second-order Taylor expansion of positon $p(t)$ and first-order expansion of velocity $v(t)$ with respect to time t [26]. Since we assume the relationships of position, velocity and acceleration to be $\dot{p}(t) = v(t)$ and $\ddot{p}(t) = \dot{v}(t) = u(t)$, then the linear dynamics take the form

$$\begin{bmatrix} p_{k+1,1} \\ p_{k+1,2} \end{bmatrix} = \begin{bmatrix} p_{k,1} \\ p_{k,2} \end{bmatrix} + \Delta t \begin{bmatrix} v_{k,1} \\ v_{k,2} \end{bmatrix} + \frac{1}{2}\Delta t^2 \begin{bmatrix} u_{k,1} \\ u_{k,2} \end{bmatrix},$$
$$\begin{bmatrix} v_{k+1,1} \\ v_{k+1,2} \end{bmatrix} = \begin{bmatrix} v_{k,1} \\ v_{k,2} \end{bmatrix} + \Delta t \begin{bmatrix} u_{k,1} \\ u_{k,2} \end{bmatrix},$$

so combining these gives the state dynamics

$$
\begin{aligned}
x_{k+1} = \begin{bmatrix} p_{k+1,1} \\ p_{k+1,2} \\ v_{k+1,1} \\ v_{k+1,2} \end{bmatrix} &= \begin{bmatrix} p_{k,1} \\ p_{k,2} \\ 0 \\ 0 \end{bmatrix} + \Delta t \begin{bmatrix} v_{k,1} \\ v_{k,2} \end{bmatrix} + \frac{1}{2}\Delta t^2 \begin{bmatrix} u_{k,1} \\ u_{k,2} \end{bmatrix} \\
&\qquad\qquad \begin{bmatrix} v_{k,1} \\ v_{k,2} \end{bmatrix} \qquad\quad \Delta t \begin{bmatrix} u_{k,1} \\ u_{k,2} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{k,1} \\ p_{k,2} \\ v_{k,1} \\ v_{k,2} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} u_{k,1} \\ u_{k,2} \end{bmatrix} \\
&= Ax_k + Bu_k \qquad (2.1c)
\end{aligned}
$$

where as previously mentioned, $p_k, v_k, u_k \in \mathbb{R}^2$ and Δt represents our discrete time step. in which

$$
A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix},
$$

If extended to 3-d space ($n = 3$), the matrix dimensions would be $A \in \mathbb{R}^{6\times 6}$ and $B_k \in \mathbb{R}^{6\times 3}$.

2.2.3 State and Control Bounds

We consider only convex sets of feasible controls and velocities, meaning that there is no minimum absolute speed v_{min} or acceleration a_{min}. Established methods do exist for dealing with non-convex control constraints via "lossless convexification" [27] [26], but these are beyond the scope of this thesis. Thus, the convex velocity and acceleration bounds take the form

$$\frac{1}{2}\left\|v_k\right\|^2 \leq \frac{1}{2}v_{max}(k)^2, \tag{2.1d}$$

$$\frac{1}{2}\|u_k\|^2 \leq \frac{1}{2}a_{max}(k)^2. \tag{2.1e}$$

2.2.4 Obstacle Avoidance

While in reality obstacles may be non-convex, they can typically be approximated and/or decomposed as the union of convex obstacles [44]. Thus, we assume that the obstacles $\mathcal{O}_{(m)}$ are convex sets with non-empty relative interior, and can be represented

$$\mathcal{O}_{(m)} = \{p \in \mathcal{W} : ||p - c_m(k)|| \leq r_m(k)\};$$

where $c_m(k) \in \mathcal{W}$ and $r_m(k)$ are, respectively, the center and radius of obstacle m at the k^{th} time step, and $\|\cdot\|$ is the L_2 Euclidean norm. This is a simplified circular representation of an obstacle of general shape and size as described in [44]. By modeling obstacles as balls rather than polygons, we can avoid the use of polytopic

constraints that are often dealt with via binary variables in mixed integer programs [32]. As depicted in Figure 2.1, the obstacles $\mathcal{O}$ restrict subsets of the state space $\mathcal{X}$, introducing non-convexity via an obstacle-avoidance constraint

$$\frac{1}{2}\left\|p_k - c_m(k)\right\|^2 \geq \frac{1}{2} r_m(k)^2, \tag{2.1f}$$

with a concave, left-hand side with respect to the vehicle position due to being bounded below.

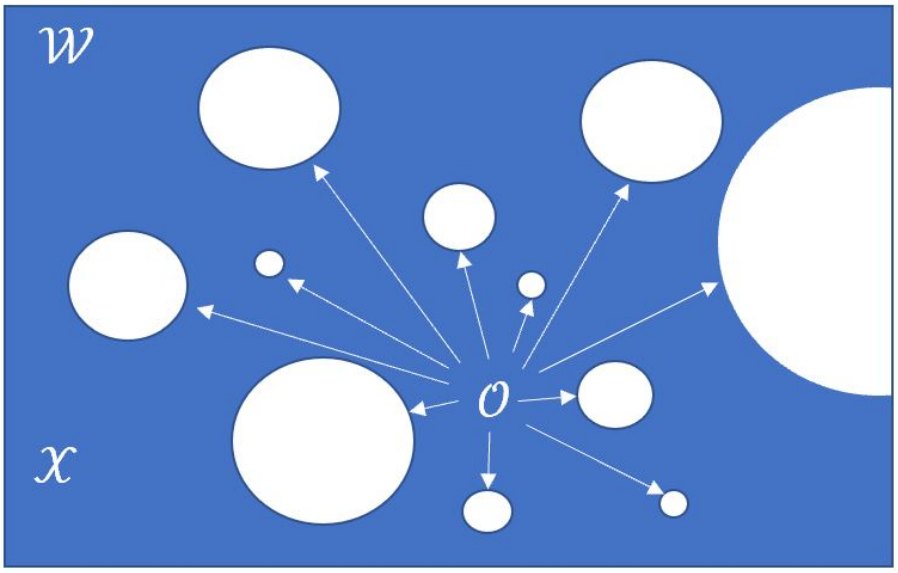


Figure 2.1: UUV Workspace with Circular Obstacles

2.2.5 Optimal Trajectory Problem

By combining our objective function, linear dynamics, state and control constraints, the discrete-time, finite-horizon, optimal control problem is given by the

constrained non-convex nonlinear program

$$\min_{u} \quad J(x,u) \triangleq \sum_{k=0}^{N} f_G(x_k,u_k) = \frac{1}{2}\sum_{k=0}^{N} \|p_G(k) - p_k\|^2 \tag{2.1a}$$

s.t.

$$x_0 = x_{IC} \tag{2.1b}$$

$$x_{k+1} = A_k x_k + B_k u_k \quad \forall \ k = 0, ..., N-1 \tag{2.1c}$$

$$\frac{1}{2}\left\|v_k\right\|^2 \leq \frac{1}{2} v_{max}(k)^2 \quad \forall \ k \in K \tag{2.1d}$$

$$\frac{1}{2}\|u_k\|^2 \leq \frac{1}{2} a_{max}(k)^2 \quad \forall \ k = 0, ..., N-1 \tag{2.1e}$$

$$\frac{1}{2}\left\|p_k - c_m(k)\right\|^2 \geq \frac{1}{2} r_m(k)^2 \quad \forall \ k \in K, m \in M \tag{2.1f}$$

where the parameters v_{max}, a_{max}, c_m, and r_m are subject to change over time, $K = \{0, ..., N\}$ and $M = \{1, ..., N_{obs}\}$. Changes in v_{max} and a_{max} may be due to water conditions (density, temperature, etc.) or imposed by a human operator, while changes in obstacle size r_m and position c_m can be updated based on improvements in resolution as the vehicle maps its environment. We address changes in the latter parameters in Chapter 4, when we investigate robust obstacles. However, throughout this chapter, we consider only stationary targets (i.e., $p_G(k) = p_G, \forall k$), static obstacles (i.e. $c_m(k) = c_m, \forall k$ and $r_m(k) = r_m, \forall k$), and constant bounds on velocity ($v_{max}(k) = v_{max}, \forall k$) and acceleration ($a_{max}(k) = a_{max}, \forall k$). Note that this formulation differs slightly from the majority of the literature, in that it does not impose a final state constraint $x_N = x_F$ as is typically done in optimal control problems. This is an important nuance, in that unlike the minimum-fuel problem $f_u(x_k, u_k) = \|u_k\|$

or minimum-time problem $f_T(x_k, u_k) = 1$ as in [25], there is no guarantee that the vehicle reaches its destination state. For reference, f_u determines an optimal solution with respect to energy consumption, while f_T finds the shortest feasible travel time t_N to reach a final state x_F.

In its current form (a quadratically-constrained, quadratic program), the optimal trajectory problem (2.1) can be solved by using many standard constrained nonlinear program solvers such as fmincon in MATLAB, CPLEX in AIMMS, or CONOPT in GAMS, though only necessarily to local optima, as has been covered extensively in the current literature [44]. Furthermore, the quality and feasibility of this solution is known to be highly dependent on an initial guess, which can sometimes be difficult to compute. While not ideal since it does not offer an obstacle-free initial trajectory, we implement an elementary, yet efficient, "warm-start" by solving (2.1) without the non-convex obstacle constraints (2.1f), which renders the problem convex. Algorithm 1 describes the sequence by which the trajectory optimization NLP (2.1) finds a trajectory from initial state x_{IC} to its target p_G. We now turn our attention to game theory to find equilibrium solution(s) to the same problem, which will allow us to consider multiple, non-cooperative players.

2.3 Game Theory Background

The focal area of this research is to generate a simulated "perfect-information reality," against which current UUV path-planning models can be validated. Game theory is ideal for this concept, since it can be used to simulate multiple indepen-

Algorithm 1 Goal-Focused Trajectory NLP

Input: initial state x_{IC}, target point p_G, max velocity v_{max}, max acceleration a_{max}, obstacle data $\mathcal{O}$, dynamics matrices A and B, step size Δt, max steps N.
Output: x, u

while $goal = 0$ **do**
  Solve (2.1) without obstacle constraint (2.1f)
  $(x, u) \leftarrow (\bar{x}, \bar{u})$ ▷ set initial guess
  Solve full NLP (2.1) with obstacle avoidance
  for $k \leftarrow 0$ to N **do**
    if $\|p_{(k)} - p_G\| < \varepsilon_G$ **then**
      $goal = 1, F = k$
      $(x, u) \leftarrow (x_0^T, ..., x_F^T, u_0^T, ..., u_F^T)^T$ ▷ Completion time step $F \leq N$
    else
      $N = N + 1$ ▷ Increase time steps and re-solve
    end if
  end for
end while

dently controlled vehicles, and whose decisions and rewards can be impacted by current or previous decisions made by all other vehicles in the system, or a subset thereof.

2.3.1 Game Theory Overview

We use game theory to model multiple agents, each acting in its own best interest (i.e., non-cooperatively), subject to the dynamics, state and control constraints described above, but whose payoff and feasible region also depends on the strategies taken by some or all of the other players. The terms "vehicle," "agent," and "player" are used interchangeably throughout this thesis.

Prior to converting the UUV trajectory problem into a game, we need to define players, strategies and outcomes. Players are individual decision makers (individuals, teams, nations, or vehicles/vessels in this case) who can take action

based on a set of possible strategies. In this thesis, we denote the set of $N_{\mathcal{I}}$ players

$$\mathcal{P} = \{\mathcal{P}_1, ...\mathcal{P}_i, ..., \mathcal{P}_{N_{\mathcal{I}}}\}$$

where each player has a set of possible strategies given by $\mathcal{S}_i$.

The particular strategy taken by player $\mathcal{P}_i$ for a single horizon is denoted $s^{[i]} \in \mathcal{S}_i$. For the current trajectory optimization problem, the strategy for $\mathcal{P}_i$ is defined by

$$s^{[i]} := (x^{[i]}, u^{[i]}),$$

where $(x^{[i]}, u^{[i]}) \in \mathcal{X} \times \mathcal{U}$ represents the state and control vectors as defined previously. The set of joint strategies $\mathcal{S} = \prod_{i \in \mathcal{I}} \mathcal{S}_i$, where $\prod$ is the Cartesian product, is a continuous set of possibilities. Given the infinite state space inherent to underwater navigation, these sets of strategies become extremely large and impossible to enumerate explicitly.

Modifying the notation from (2.1), the objective, or payoff, for player $\mathcal{P}_i$ is now $J^{[i]}(s)$ if $s \in \mathcal{S}$ is the joint strategy chosen by all players.

In a single-stage equilibrium, each independent agent, or player, knows exactly the state and control sequence of not only itself, but of all other players that influence its objective across the entire time horizon. This changes, however, when we consider the rolling-horizon case in Chapter 4, in which players can exhibit learning behavior through updated policies in their respective objective functions.

2.3.2 Nash Equilibrium

A set of joint strategies $(\bar{s}_1, ..., \bar{s}_i, ..., \bar{s}_{N_\mathcal{I}}) \in \prod_{i \in \mathcal{I}} \mathcal{S}_i$ is said to be a Nash Equilibrium (NE) if no player would find it beneficial to deviate provided that all other players do not deviate from their strategies played at the NE outcome. Formally, for every player $\mathcal{P}_i \in \mathcal{P}$, in which the cost functional $J^{[i]}$ is to be minimized,

$$J^{[i]}(\bar{s}_i, \bar{s}_{-i}) \leq J^{[i]}(s_i, \bar{s}_{-i}) \ \ \forall s_i \in \mathcal{S}_i,$$

where s_{-i} represents the strategies of all players other than $\mathcal{P}_i$ [16]. In other words, $\bar{s}_i \in \mathcal{S}_i$ is a Nash equilibrium if and only if, for every player i, $\bar{s}_i \in \mathcal{S}_i$ is the best-case strategy when $\bar{s}_{-i} \in \mathcal{S}_{-i}$ [16]. We leverage the Karush-Kuhn-Tucker optimality conditions for each player's optimization problem to find such an equilibrium.

2.3.3 Karush-Kuhn-Tucker Conditions

To find equilibrium points across the planning horizon, we can first put the problem in standard form

$$\min_x \quad f(x) \tag{2.2a}$$

$$\text{s.t} \quad g_i(x) \leq 0 \quad \forall i \in I = 1, ..., m \quad (\lambda_i) \tag{2.2b}$$

$$h_j(x) = 0 \quad \forall j \in J = 1, ..., p \quad (\mu_j) \tag{2.2c}$$

where λ_i and μ_j are the Lagrange multipliers for the i^{th} inequality constraint and j^{th} equality constraint, respectively [4]. A KKT point is a triple (x^*, λ^*, μ^*) where

$x^* \in R^n$, $\lambda^* \in R^m$ and $\mu^* \in R^p$ such that the following conditions hold

$$\nabla f(x^*) + \sum_i^m \lambda_i^* \nabla g_i(x^*) + \sum_j^p \mu_j^* \nabla h_j(x^*) = 0 \tag{2.3a}$$

$$g_i(x^*) \leq 0, \quad \lambda_i^* \geq 0, \quad \lambda_i^* g_i(x^*) = 0 \qquad \forall \ i = 1, ..., m \tag{2.3b}$$

$$h_j(x^*) = 0, \quad \mu_j^* \ \text{free} \qquad \forall \ j = 1, ..., p \tag{2.3c}$$

The KKT conditions are necessary if a constraint qualification holds [2]. They are sufficient, for example, if objective function f, given g_i is convex, and h_j is affine [2].

Chapter 3

Nash Equilibrium Solutions

3.1 Single-Vehicle Trajectory Equilibrium

Now we can apply the equilibrium methodology to the UUV formulation, first by putting the problem in standard form as described in (2.3). Note that in our original problem, (2.1a)-(2.1f) are intentionally squared and halved to simplify the next steps in finding KKT points without loss of accuracy in the program due to the norms being non-negative, monotonically increasing functions. In standard form, the UUV formulation becomes

$$\min_{u} \quad J(x,u) \triangleq \sum_{k=0}^{N} f_G(x_k,u_k) = \frac{1}{2}\sum_{k=0}^{N} \left\| p_G - p_k \right\|^2 \tag{3.1a}$$

s.t.

$$h_0(x_0,u_0) \triangleq x_0 - x_{IC} = 0 \quad (\mu_0) \tag{3.1b}$$

$$h_k(x,u) \triangleq x_k - Ax_{k-1} - Bu_{k-1} = 0 \quad \forall\ k = 1,...,N \quad (\mu_k) \tag{3.1c}$$

$$g_k^{vel}(x_k) \triangleq \frac{1}{2}\left(\left\| v_k \right\|^2 - v_{max}^2 \right) \leq 0 \quad \forall\ k \in K \quad (\sigma_k) \tag{3.1d}$$

$$g_k^{acc}(u_k) \triangleq \frac{1}{2}\left(\left\| u_k \right\|^2 - a_{max}^2 \right) \leq 0 \quad \forall\ k = 0,...,N-1 \quad (\nu_k) \tag{3.1e}$$

$$g_{km}^{obs}(x_k) \triangleq \frac{1}{2}\left(r_m^2 - \left\| p_k - c_m \right\|^2 \right) \leq 0 \quad \forall\ k \in K, m \in M \quad (\lambda_{km}) \tag{3.1f}$$

where the sets K and M represent the time and obstacle indices, respectively. The Lagrange multipliers are represented by the Greek letters contained in $(\cdot)$ adjacent to their corresponding primal constraints. The inequality multipliers are all one-dimensional, but $\mu_k \in \mathbb{R}^4$ is a vector of the same dimensions as the state vector x_k for the linear equality constraints. Meanwhile, since the cost functional $J(x, u)$ depends on both the state vector x and control vector u, the multipliers for (3.1a) $\forall k \in K$ are $x_k \in \mathbb{R}^4$ and $u_k \in \mathbb{R}^2$.

3.1.1 Trajectory Problem in Expanded Vector Form

We can expand the squared norms found in (3.1) and retain the position and velocity components from the state vector to most clearly illustrate the example to follow.

$$
\begin{aligned}
f_G(x_k, u_k) &= \frac{1}{2}\left\|p_G - x_k\right\|^2 = \frac{1}{2}(p_G - p_k)^T(p_G - p_k) \\
&= \frac{1}{2}[p_G^T p_G - p_G^T p_k - p_k^T p_G + p_k^T p_k] \\
&= -p_G^T I p_k + \frac{1}{2}(p_k^T I p_k + p_G^T I p_G), \qquad (3.2a)
\end{aligned}
$$

$$
h(x, u) \triangleq \mathbf{A}_h x + \mathbf{B}_h u - b = 0 \qquad (3.2b)
$$

$$
g_k^{vel}(x_k) = \frac{1}{2}\left(\left\|v_k\right\|^2 - v_{max}^2\right) = \frac{1}{2}\left(v_k^T I v_k - v_{max}^2\right) \qquad (3.2c)
$$

$$
g_k^{acc}(u_k) = \frac{1}{2}\left(\|u_k\|^2 - a_{max}^2\right) = \frac{1}{2}\left(u_k^T I u_k - a_{max}^2\right) \qquad (3.2d)
$$

$$
\begin{aligned}
g_{km}^{obs}(x_k) &= \frac{1}{2}\left(r_m^2 - \left\|p_k - c_m\right\|^2\right) = \frac{1}{2}\left(r_m^2 - (p_k - c_m)^T I (p_k - c_m)\right) \\
&= \frac{1}{2}\left(r_m^2 - p_k^T I p_k + p_k^T I c_m + c_m I p_k - c_m^T c_m\right) \\
&= \frac{1}{2}\left(r_m^2 - c_m^T I c_m - p_k^T I p_k\right) + p_k^T I c_m \qquad (3.2e)
\end{aligned}
$$

where $\mathbf{A}_h$ is a square block-diagonal matrix of rank $2n(N+1)$ that represents the state coefficients from (3.1c), in which the identity matrix $I_{4\times4}$ is the implied coefficient for x_k, and $\mathbf{B}_h$ is a non-square $2n(N+1) \times n(N+1)$ matrix providing the control coefficients, where $0_{4\times2}$ is the implied coefficient for u_k. Thus, $\mathbf{A}_h$ and $\mathbf{B}_h$ take the form

$$\mathbf{A}_h = \begin{bmatrix} I_{4\times4} & 0_{4\times4} & \cdots & 0_{4\times4} \\ -A & I_{4\times4} & \ddots & \vdots \\ 0_{4\times4} & \ddots & \ddots & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & -A & I_{4\times4} \end{bmatrix}, \quad \mathbf{B}_h = \begin{bmatrix} 0_{4\times2} & 0_{4\times2} & \cdots & 0_{4\times2} \\ -B & 0_{4\times2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0_{4\times2} \\ 0_{4\times2} & \cdots & -B & 0_{4\times2} \end{bmatrix}. \tag{3.3}$$

where the submatrices $-A$ and $-B$ are the negatives of the 4×4 A and 4×2 B matrices defined in Section 2.2. The vector $b = [x_{IC}^T, 0, ..., 0]^T$ includes the initial condition from equation (3.1b), which fixes the initial state x_0 to the known initial position and velocity at $k = 0$. Given a function that has the form $f(x) = \frac{1}{2}x^T Mx$ whose matrix $M = M^T$ (symmetric), it can be shown that $\nabla f(x) = Mx$. Also, given the linear nature of the dynamic equations, we know that $\nabla_x h = \mathbf{A}^T$, while $\nabla_u h = \mathbf{B}^T$. Therefore our problem has the following KKT conditions:

$\forall$ time steps $k \in K$:

$$\nabla_k L(x,u) \triangleq \nabla f_G(x_k,u_k) + \nabla h_k(x_k,u_k)\mu_k + \sigma_k \nabla g_k^{vel}(x_k,u_k) \tag{3.4a}$$

$$+ \nu_k \nabla g_k^{acc}(x_k,u_k) + \sum_{m\in M} \lambda_{km} \nabla g_{km}^{obs}(x_k,u_k) = 0 \ , \ x_k, u_k \text{ free}$$

$$h_k(x,u) = 0 \ , \ \mu_k \text{ free} \quad . \tag{3.4b}$$

$$g_k^{vel}(x_k,u_k) \leq 0 \ \perp \ \sigma_k \geq 0 \tag{3.4c}$$

$$g_k^{acc}(x_k,u_k) \leq 0 \ \perp \ \nu_k \geq 0 \tag{3.4d}$$

$$g_{km}^{obs}(x_k,u_k) \leq 0 \ \perp \ \lambda_{km} \geq 0 \quad \forall \ m \in M \tag{3.4e}$$

Suppose we have a function $f := f(x_k, u_k)$ that is continuous and differentiable with respect to both x_k and u_k. Although some components of f may only explicitly be a function of the state vector and others a function of the control, for uniformity and clarity, we can define $\nabla f(x_k, u_k)$ to be the component-wise partial derivative with respect to $x_k = (p_k^T, v_k^T)^T$ and u_k for each time step k

$$\nabla f := \begin{bmatrix} \nabla_{p_k} f(p_k, v_k, u_k) \\ \nabla_{v_k} f(p_k, v_k, u_k) \\ \nabla_{u_k} f(p_k, v_k, u_k) \end{bmatrix}$$

where $\nabla f \in \mathcal{X} \times \mathcal{U} \subseteq \mathbb{R}^6$. We will apply this to an illustrative example below.

3.1.2 KKT Points Derivation

To construct the Lagrangian derivative (3.4a), we now address each term. The gradient of the "goal" objective function (3.2a) is

$$\nabla_k f_G = \begin{bmatrix} \nabla_{x_k}[-p_G^T I p_k + \frac{1}{2}(p_k^T I p_k + p_G^T I p_G)] \\ \nabla_{u_k}[-p_G^T I p_k + \frac{1}{2}(p_k^T I p_k + p_G^T I p_G)] \end{bmatrix} = \begin{bmatrix} -p_G + p_k \\ 0 \\ 0 \end{bmatrix} \quad \forall k \in K, \quad (3.5a)$$

so then the non-zero size of $\nabla_k f_G$ is of the same rank as $p_k \in \mathbb{R}^2$. Now, the gradients of the vehicle dynamics constraints (3.4b) need to be taken over the entire time horizon since the variables of the previous time step $k-1$ directly affect the

variables in time step k

$$\nabla h = \begin{bmatrix} \nabla_x[\mathbf{A}_h x + \mathbf{B}_h u - b] \\ \nabla_u[\mathbf{A}_h x + \mathbf{B}_h u - b] \end{bmatrix} = \begin{bmatrix} \mathbf{A}_h^T \\ \mathbf{B}_h^T \end{bmatrix},$$

where we can parse out the sub-matrices A^T and B^T to identify the gradients at each time step k. If we let ${\mathbf{A}_h}^T = \begin{bmatrix} I_{4\times4} & -A^T \end{bmatrix}$, and ${\mathbf{B}_h}^T = \begin{bmatrix} 0_{2\times4} & -B^T \end{bmatrix}$, where only the blocks associated with time steps k and $k+1$ are included in these sub-matrices. Unlike the other Lagrange multipliers, $\mu_k \in \mathbb{R}^4$ is a vector of the same rank as x_k, where we can identify the subcomponents $\mu_k = ((\mu_k^p)^T, (\mu_k^v)^T)^T$, while $\nabla h_k \in \mathbb{R}^{6\times8}$. This gives us, $\forall k = 0, ..., T-1$,

$$\nabla h_k \mu_k = \begin{bmatrix} I_{4\times4} & -A^T \\ 0_{2\times4} & -B^T \end{bmatrix} \begin{bmatrix} \mu_k^p \\ \mu_k^v \\ \mu_{k+1}^p \\ \mu_{k+1}^v \end{bmatrix} = \begin{bmatrix} \mu_k^p - \mu_{k+1}^p \\ \mu_k^v - \mu_{k+1}^v - \Delta t \mu_{k+1}^p \\ -\frac{1}{2}\Delta t^2 \mu_{k+1}^p - \Delta t \mu_{k+1}^v \end{bmatrix} \tag{3.5b}$$

and for the last time step T, $\nabla h_N \mu_N = [(\mu_N^p)^T, (\mu_N^p)^T, 0]^T$.

The other constraints are more straight-forward, since there is no interaction across time steps, and they are all scalars, rather than vectors. Thus, taking the

gradient of (3.2c) and (3.2d), respectively, we have

$$\sigma_k \nabla g_k^{vel} = \sigma_k \begin{bmatrix} \nabla_{x_k} \frac{1}{2} \left(v_k^T I v_k - v_{max}^2 \right) \\ \nabla_{u_k} \frac{1}{2} \left(v_k^T I v_k - v_{max}^2 \right) \end{bmatrix} = \sigma_k \begin{bmatrix} 0 \\ v_k \\ 0 \end{bmatrix} \quad \forall k \in K, \tag{3.5c}$$

$$\nu_k \nabla g_k^{acc} = \nu_k \begin{bmatrix} \nabla_{x_k} \frac{1}{2} \left(u_k^T I u_k - a_{max}^2 \right) \\ \nabla_{u_k} \frac{1}{2} \left(u_k^T I u_k - a_{max}^2 \right) \end{bmatrix} = \nu_k \begin{bmatrix} 0 \\ 0 \\ u_k \end{bmatrix} \quad \forall k = 0, ..., T-1. \tag{3.5d}$$

Finally, taking the gradient of the obstacle constraint (3.2e) gives us

$$\begin{aligned} \lambda_{km} \nabla g_k^{obs} &= \lambda_{km} \begin{bmatrix} \nabla_{x_k} [\frac{1}{2} \left(r_m^2 - c_m^T c_m - p_k^T p_k \right) + p_k^T c_m] \\ \nabla_{u_k} [\frac{1}{2} \left(r_m^2 - c_m^T c_m - p_k^T p_k \right) + p_k^T c_m] \end{bmatrix} \\ &= \lambda_{km} \begin{bmatrix} c_m - p_k \\ 0 \\ 0 \end{bmatrix}, \quad \forall k \in K, m \in M \end{aligned} \tag{3.5e}$$

So replacing (3.4a) with the sum of (3.5a)-(3.5e), and combining with the complementary constraints (3.4b)-(3.4e), we can now determine the KKT point(s) for a small example.

3.1.3 Illustrative KKT Example

To illustrate this concept, we consider the two-dimensional ($n = 2$) problem with one time step ($T = 1$) in the horizon for one vehicle and one static obstacle

$(N = 1, N_{obs} = 1)$. The parameters for this system are defined in Table 3.1. The obstacle information is condensed to $\mathcal{O}_{(m)} = (c_{m,1}, c_{m,2}, r_m)^T$ for brevity.

Table 3.1: Single Vehicle Illustration Parameters

Parameter	Value	Units
t_0	0	h
Δt	1	h
T_h	1	h
x_{IC}	$(0, 0, 0, 0)^T$	km
x_G	$(9, 3, 0, 0)^T$	km
v_{max}	5	km/h
a_{max}	20	km/h^2
$\mathcal{O}_{(1)}$	$(4, 1, 2)^T$	km

To show row-wise relationships within the Lagrangian derivative equation, we temporarily separate the variable vectors into their physical dimensional components, so the state vector $x_k = [(p_{k,1})^T, (p_{k,2})^T, (v_{k,1})^T, (v_{k,2})^T)]^T$, the control vector $u_k = [(u_{k,1})^T, (u_{k,2})^T]^T$, and the dynamics Lagrange multiplier is now $\mu_k = [(\mu^p_{k,1})^T, (\mu^p_{k,2})^T, (\mu^v_{k,1})^T)^T, (\mu^v_{k,2})^T]^T$ Thus, the derivative of the Lagrangian ∇L_k at

each time step k is

$$\nabla L_0 = \begin{bmatrix} p_{0,1} - 9 \\ p_{0,2} - 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \mu^p_{0,1} - \mu^p_{1,1} \\ \mu^p_{0,2} - \mu^p_{1,2} \\ \mu^v_{0,1} - \mu^v_{1,1} - \Delta t \mu^p_{1,1} \\ \mu^v_{0,2} - \mu^v_{1,2} - \Delta t \mu^p_{1,2} \\ -\frac{1}{2}\Delta t^2 \mu^p_{1,1} - \Delta t \mu^v_{1,1} \\ -\frac{1}{2}\Delta t^2 \mu^p_{1,2} - \Delta t \mu^v_{1,2} \end{bmatrix} + \sigma_0 \begin{bmatrix} 0 \\ 0 \\ v_{0,1} \\ v_{0,2} \\ 0 \\ 0 \end{bmatrix} + \nu_0 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u_{0,1} \\ u_{0,2} \end{bmatrix} + \lambda_1 \begin{bmatrix} 4 - p_{0,1} \\ 1 - p_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

$$\nabla L_1 = \begin{bmatrix} p_{1,2} - 9 \\ p_{1,2} - 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \mu^p_{1,1} \\ \mu^p_{1,2} \\ \mu^v_{1,1} \\ \mu^v_{1,2} \\ 0 \\ 0 \end{bmatrix} + \sigma_1 \begin{bmatrix} 0 \\ 0 \\ v_{1,1} \\ v_{1,2} \\ 0 \\ 0 \end{bmatrix} + \nu_1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u_{1,1} \\ u_{1,2} \end{bmatrix} + \lambda_1 \begin{bmatrix} 4 - p_{1,1} \\ 1 - p_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

The linear dynamics constraints $h_k(x, u) = x_k - Ax_{k-1} - Bu_{k-1} - b_k$, each associ-

ated with a free dual variable vector μ_k for $k = 0, 1$ are

$$h_0 = \begin{bmatrix} p_{0,1} \\ p_{0,2} \\ v_{0,1} \\ v_{0,1} \end{bmatrix} - \begin{bmatrix} p_{IC,1} \\ p_{IC,2} \\ v_{IC,1} \\ v_{IC,2} \end{bmatrix} = 0$$

$$h_1 = \begin{bmatrix} p_{1,1} \\ p_{1,2} \\ v_{1,1} \\ v_{1,2} \end{bmatrix} - \begin{bmatrix} p_{0,1} + \Delta t v_{0,1} \\ p_{0,2} + \Delta t v_{0,2} \\ v_{0,1} \\ v_{0,2} \end{bmatrix} - \begin{bmatrix} \frac{1}{2}\Delta t u_{0,1} \\ \frac{1}{2}\Delta t u_{0,2} \\ \Delta t u_{0,1} \\ \Delta t u_{0,2} \end{bmatrix} = 0$$

The velocity and acceleration bound complementarity equations are

$$\begin{aligned} \frac{1}{2}\sigma_0 \left[(v_{0,1})^2 + (v_{0,2})^2 - 5^2\right] &= 0 \\ \frac{1}{2}\sigma_1 \left[(v_{1,1})^2 + (v_{1,2})^2 - 5^2\right] &= 0 \\ \frac{1}{2}\nu_0 \left[(u_{0,1})^2 + (u_{0,2})^2 - 20^2\right] &= 0 \\ \frac{1}{2}\nu_1 \left[(u_{1,1})^2 + (u_{1,2})^2 - 20^2\right] &= 0 \end{aligned}$$

Finally, the obstacle avoidance complementarity equations are

$$\begin{aligned} \lambda_0 \left[\frac{1}{2}\left(2^2 - (4^2 + 1^2) - (p_{0,1})^2 - (p_{0,2})^2\right) + 4p_{0,1} + 1p_{0,2}\right] &= 0 \\ \lambda_1 \left[\frac{1}{2}\left(2^2 - (4^2 + 1^2) - (p_{1,1})^2 - (p_{1,2})^2\right) + 4p_{1,1} + 1p_{1,2}\right] &= 0 \end{aligned}$$

3.1.4 Solving as a Mixed Complementarity Problem

We can solve such problems, as laid out in Section 3.1.3, as mixed complementarity problems (MCP), which generalize a system of nonlinear equations determined through a nonlinear function $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$ with lower and upper bounds on variables [3], where an MCP for $x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}$ has the structure

$$0 \leq F_x(x, y) \quad \perp \quad x \geq 0$$

$$0 = F_y(x, y) \ , \ \ y \ \text{ free.}$$

Taking the notation from (2.3), the problem can be recast as an MCP of the form:

$$z = \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix}, \quad F(z) = \begin{bmatrix} \nabla f(x) + \lambda \nabla g(x) + \mu \nabla h(x) \\ -g(x) \\ h(x) \end{bmatrix} \tag{3.6}$$

A key requirement to solving these types of problems to equilibrium as an MCP is that the the problem is square (i.e., number of variables and number of equations matches). For instance, the data in the illustrative example above includes eight state variables $p_{0,1}, p_{0,2}, v_{0,1}, v_{0,2}, p_{1,1}, p_{1,2}, v_{1,1}, v_{1,2} \subseteq x$ and four control variables $u_{0,1}, u_{0,2}, u_{1,1}, u_{1,2} \subseteq u$ with six inequality constraints (i.e. six multipliers) and eight equality constraints (i.e. eight multipliers), for a total of 26 variables and 26 equations. We use the PATH solver in GAMS to solve these MCPs, in addition to a MATLAB-GAMS interface.

Figure 3.1 shows the path taken by the vehicle in the problem with simple parameters where acceleration is neglected (i.e., the control matrix B is removed from the dynamics constraints). This example provides visual validation of the KKT solution procedure presented in this section. Recall the most well-known "Pythagorean triple," a right triangle with edge lengths having a ratio of 3-4-5 (e.g., two legs of length 3 and 4, and hypotenuse of length 5) [43]. Given the initial point $p_0 = (0, 0)^T$, target point $p_G = (9, 3)^T$ and maximum velocity $v_{max} = 5$ m/s, with the obstacle centered at $c_1 = (4, 1)$ and radius $r_1 = 2$, the reader can employ the "3-4-5 rule" to observe the shortest path with obstacle-avoidance at the collocation points over two time steps is uniquely $p_0 = (0, 0)^T, p_1 = (4, 3)^T, p_2 = (9, 3)^T$. Note

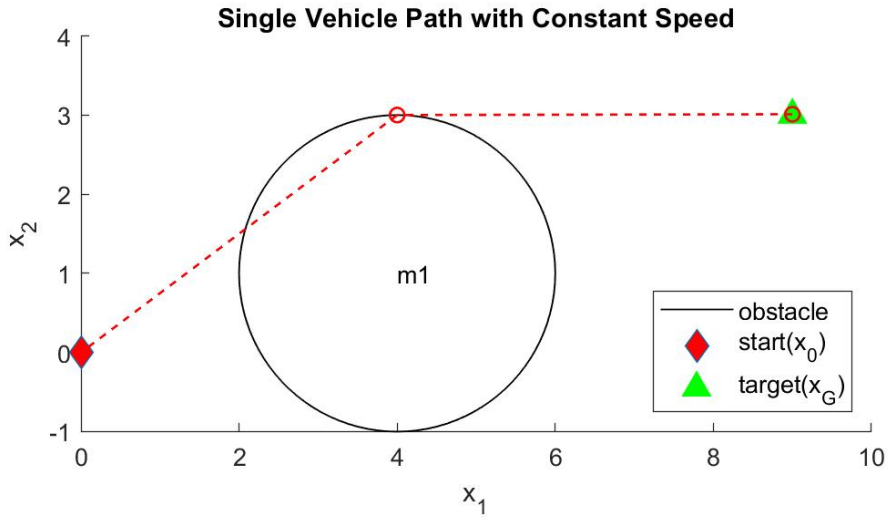


Figure 3.1: Single vehicle path with two time steps

that the path violates the obstacle constraint between $t = 0$ and $t = 1$, which is

mitigated with shorter time steps ($\Delta t = 0.1$) and more evaluations, as shown in Figure 3.2. However, this increase in the number of time steps can quickly make the problem intractable since we add six primal variables and at least seven (depends on number of obstacles) constraints for every time step. Now, this does not guarantee

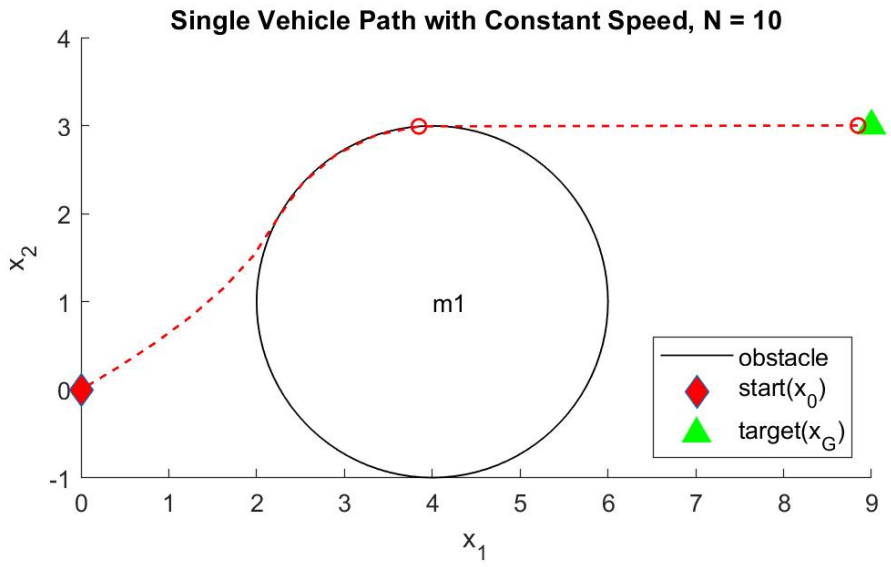


Figure 3.2: Single vehicle path with 10 time steps

a unique solution, but rather a local optimum when solved for a single vehicle.

3.2 Extension to the Pursuit-Evasion Game

Now that we have the tools to solve equilibrium problems, we can investigate a somewhat more complex problem in addressing multi-agent, non-cooperative games. Prior to formally defining our equilibrium problem, some useful notation and definitions must be introduced. Let $\mathcal{A}$ denote the set of attacker UUV(s), and

$\mathcal{D}$ denote the set of defender UUV(s). Thus, our player list is split such that

$$\mathcal{P}_{\mathcal{A}} = \{\mathcal{P}_1, ...\mathcal{P}_i, ..., \mathcal{P}_{|\mathcal{P}_{\mathcal{A}}|}\}, \quad \mathcal{P}_{\mathcal{D}} = \{\mathcal{P}_{|\mathcal{P}_{\mathcal{A}}|+1}, ...\mathcal{P}_i, ..., \mathcal{P}_{N_{\mathcal{I}}}\}, \quad \mathcal{P}_{\mathcal{A}}, \mathcal{P}_{\mathcal{D}} \subseteq \mathcal{P},$$

$$\mathcal{P}_{\mathcal{A}} \cap \mathcal{P}_{\mathcal{D}} = \emptyset, \quad \mathcal{P}_{\mathcal{A}} \cup \mathcal{P}_{\mathcal{D}} = \mathcal{P},$$

where $i = \{1, ..., N_{\mathcal{I}}\}$ is the vehicle index. For now, let $|\mathcal{P}_{\mathcal{A}}| = |\mathcal{P}_{\mathcal{D}}| = 1$, meaning we only consider a two-player, non-cooperative game where $\mathcal{P}_{\mathcal{A}} = \mathcal{P}_1$ and $\mathcal{P}_{\mathcal{D}} = \mathcal{P}_2$. This also requires an update in our variable and constraint notation, since both agents are defined by states in the same space $x_k^{[1]}, x_k^{[2]} \in \mathcal{X}_k$, and are controlled in the same space $u_k^{[1]}, u_k^{[2]} \in \mathcal{U}_k$ for each time step k. Each vehicle is independently subject to the same set of constraints described in (2.1). However, it is necessary to modify their cost functions $J^{[1]}(s)$ and $J^{[2]}(s)$, where recall $s = (s^{[1]}, s^{[2]}) \in \mathcal{S}$ is the joint strategy chosen by all players.

3.2.1 Pursuit Function and Multi-Objective Equilibrium

The case study addressed in this thesis involves an attacker UUV $\mathcal{P}_1$ whose mission it is to reach a stationary target position p_G as quickly as possible, and a defender UUV $\mathcal{P}_2$ whose mission it is to intercept said attacker before reaching the target. This variant of the "pursuit-evasion game" (PEG) pictured in Figure 3.3 is inspired partially from the multi-pursuer game described in [42] and defender-attacker-target game described in [41]. First, to account for multiple players simultaneously, the variable notation is updated so that $\mathcal{P}_i$'s position at time step k in dimension d is given by $p_{d,k}^{[i]}$, and likewise for its velocity $v_{d,k}^{[i]}$ and acceleration $u_{d,k}^{[i]}$.

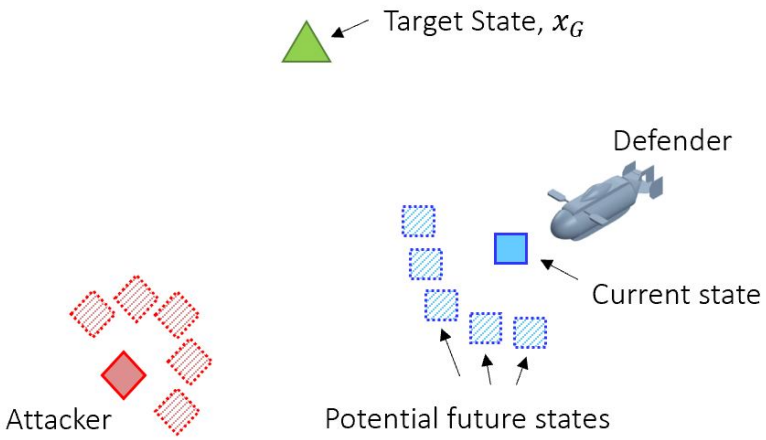


Figure 3.3: Pursuit-Evasion Game Scenario

We define the "pursuit function", f_P which serves as the defender $\mathcal{P}_2$'s objective function for each time step k

$$f_P(x_k^{[1]}, u_k^{[1]}, x_k^{[2]}, u_k^{[2]}) = \frac{1}{2}\|p_k^{[2]} - p_k^{[1]}\|^2 \tag{3.7a}$$

which the defender tries to minimize. The reader should note that this objective function defines the defender more as a "surveillance asset" focused solely on the attacker vehicle, rather than actually defending a particular point or area, as the name implies. This results sometimes in the defender following behind the attacker rather than taking the shortest path to intercept. Since the constraints are independently enforced, the defender's optimization problem minimizes (3.7a) subject

to (2.1b)-(2.1f), taking the form

$$\min_{u^{[2]}} \quad J^{[2]}(s) \triangleq \sum_{k=0}^{N} f_P(x_k^{[1]}, u_k^{[1]}, x_k^{[2]}, u_k^{[2]}) \tag{3.7a}$$

s.t.

$$x_0^{[2]} = x_{IC}^{[2]} \tag{3.7b}$$

$$x_{k+1}^{[2]} = Ax_k^{[2]} + Bu_k^{[2]} \quad \forall \ k = 0, ..., N-1 \tag{3.7c}$$

$$\frac{1}{2}\left\|v_k^{[2]}\right\|^2 \leq \frac{1}{2} v_{max}(k)^2 \quad \forall \ k \in K \tag{3.7d}$$

$$\frac{1}{2}\|u_k^{[2]}\|^2 \leq \frac{1}{2} a_{max}(k)^2 \quad \forall \ k = 0, ..., N-1 \tag{3.7e}$$

$$\frac{1}{2}\left\|p_k^{[2]} - c_m(k)\right\|^2 \geq \frac{1}{2} r_m(k)^2 \quad \forall \ k \in K, m \in M \tag{3.7f}$$

which is solved for the defender $\mathcal{P}_2$. Meanwhile, the attacker $\mathcal{P}_1$ now seeks to minimize a multi-objective optimization problem with weighted competing objectives. $\mathcal{P}_1$ retains the objective to reach its target point p_G using the goal function, restated from (2.1a) with new notation

$$f_G(x_k^{[1]}, u_k^{[1]}) = \frac{1}{2}\|p_G - p_k^{[1]}\|^2,$$

while it also simultaneously attempts to evade defender $\mathcal{P}_1$ (i.e., maximize the pursuit function f_P). Thus, the cost functional over the entire horizon for the attacker $\mathcal{P}_1$ is

$$\min_{u^{[1]}} \quad J^{[1]}(s) \triangleq \sum_{k=0}^{N} \left[\omega_G f_G(x_k^{[1]}, u_k^{[1]}) - \omega_P f_P(x_k^{[1]}, u_k^{[1]}, x_k^{[2]}, u_k^{[2]})\right], \tag{3.8a}$$

where $\omega_G, \omega_P \in [0, 1]$ are a weighting parameters and $\omega_G + \omega_P = 1$. The effects of these two objectives on $\mathcal{P}_1$ are depicted visually in Figure 3.4. Note that the effects

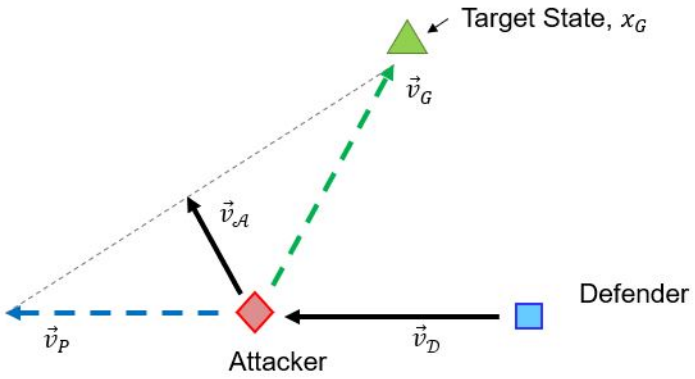


Figure 3.4: Effects of weighted sub-objectives ($\omega_G = \omega_P$) on Attacker $\mathcal{P}_1$

of each sub-objective can be interpreted as vectors of magnitude $|\vec{v}_P| = 2\sqrt{f_P}$ and $|\vec{v}_G| = 2\sqrt{f_G}$, respectively. The direction of each are determined based on relative position to the defender, $\hat{v}_P = \frac{p_k^{[1]} - p_k^{[2]}}{\|p_k^{[1]} - p_k^{[2]}\|}$, and the target, $\hat{v}_G = \frac{p_G - p_k^{[1]}}{\|p_G - p_k^{[1]}\|}$, respectively. Thus, summing the two vectors with respective coefficients gives us the cumulative effect and velocity for $\mathcal{P}_{\mathcal{A}}$

$$\vec{v}_{\mathcal{A}} = \omega_P \vec{v}_P + \omega_G \vec{v}_G.$$

The weighting parameters should be tuned based on scenario conditions, which is a topic of discussion in Chapter 4, when the attacker may prioritize the goal function over evasion if it is closer to the target than any defenders. Again, the attacker's trajectory is subject to the same constraints as previously stated.

3.2.2 Two-player Game KKT Conditions

The KKT conditions for the two-player pursuit evasion game are constructed similarly to the single-vehicle example (3.4), producing a larger system of equations that encompasses both players simultaneously, such that:

$\forall$ players $\mathcal{P}_i \in \mathcal{P}$, $\forall$ time steps $k \in K$:

$$\nabla_k L^{[i]}(s) \triangleq \nabla_k J^{[i]}(s) + \nabla h_k(x_k^{[i]}, u_k^{[i]})\mu_k^{[i]} + \sigma_k^{[i]} \nabla g_k^{vel}(x_k^{[i]}) \tag{3.9a}$$

$$+ \nu_k^{[i]} \nabla g_k^{acc}(u_k^{[i]}) + \sum_{m \in M} \lambda_{km}^{[i]} \nabla g_{km}^{obs}(x_k^{[i]}) = 0 \ , \ \ x_k, u_k \ \text{ free}$$

$$h_k(x^{[i]}, u^{[i]}) = 0 \ , \ \ \mu_k^{[i]} \ \text{ free} \quad . \tag{3.9b}$$

$$g_k^{vel}(x_k^{[i]}) \leq 0 \ \ \perp \ \ \sigma_k^{[i]} \geq 0 \tag{3.9c}$$

$$g_k^{acc}(u_k^{[i]}) \leq 0 \ \ \perp \ \ \nu_k^{[i]} \geq 0 \tag{3.9d}$$

$$g_{km}^{obs}(x_k^{[i]}) \leq 0 \ \ \perp \ \ \lambda_{km}^{[i]} \geq 0 \quad \forall \ m \in M \tag{3.9e}$$

where the cost functionals $J^{[i]}$ for $\mathcal{P}_1$ and $\mathcal{P}_2$ are the weighted multi-objective function (3.8a) and pursuit function (3.7a), respectively. The gradient of $\mathcal{P}_1$'s cost functional is

$$\nabla_k J^{[1]} = \omega_G \begin{bmatrix} p_k^{[1]} - p_G \\ 0 \\ 0 \end{bmatrix} - \omega_P \begin{bmatrix} p_k^{[1]} - p_k^{[2]} \\ 0 \\ 0 \end{bmatrix} \ \forall k \in K, \tag{3.10a}$$

while the the gradient of $\mathcal{P}_2$'s cost functional is

$$\nabla_k J^{[2]} = \begin{bmatrix} p_k^{[2]} - p_k^{[1]} \\ 0 \\ 0 \end{bmatrix} \quad \forall k \in K. \tag{3.10b}$$

The remaining terms in each player $\mathcal{P}_i$'s Lagrangian gradient (3.9a) corresponding to each constraint remain the same as described for the single-vehicle example (3.5b)-(3.5e).

3.2.3 Two-player Equilibrium Solution

This system of equations is solved simultaneously for both players to find an equilibrium solution. To illustrate this, we consider a two-player, four obstacle scenario with the parameters shown in Table 4.1. To make the equilibrium solution

Table 3.2: Two-Player PEG Parameters

Parameter	Value	Units
t_0	0	s
Δt	0.1	s
T_h	5	s
$x_{IC}^{[1]}$	$(0,0,0,0)^T$	m
$x_{IC}^{[2]}$	$(0,10,0,0)^T$	m
x_G	$(17,9,0,0)^T$	m
v_{max}	5	m/s
a_{max}	20	m/s^2
$\mathcal{O}_{(1)}$	$(4,1,2)^T$	m
$\mathcal{O}_{(2)}$	$(8,6,3)^T$	m
$\mathcal{O}_{(3)}$	$(9,0,2.5)^T$	m
$\mathcal{O}_{(4)}$	$(14,5,3)^T$	m

clear, we describe the full expanded system of equations for the initial time steps $k =$

$0, 1$, which can be extrapolated over the entire horizon. The Lagrangian derivatives for each player are:

$$\nabla_0 L^{[1]} = \omega_G \begin{bmatrix} p^{[1]}_{0,1} - 17 \\ p^{[1]}_{0,2} - 9 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \omega_P \begin{bmatrix} p^{[1]}_{0,1} - p^{[2]}_{0,1} \\ p^{[1]}_{0,2} - p^{[2]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \mu^{p}{}^{[1]}_{0,1} - \mu^{p}{}^{[1]}_{1,1} \\ \mu^{p}{}^{[1]}_{0,2} - \mu^{p}{}^{[1]}_{1,2} \\ \mu^{v}{}^{[1]}_{0,1} - \mu^{v}{}^{[1]}_{1,1} - \Delta t \mu^{p}{}^{[1]}_{1,1} \\ \mu^{v}{}^{[1]}_{0,2} - \mu^{v}{}^{[1]}_{1,2} - \Delta t \mu^{p}{}^{[1]}_{1,2} \\ -\frac{1}{2}\Delta t^2 \mu^{p}{}^{[1]}_{1,1} - \Delta t \mu^{v}{}^{[1]}_{1,1} \\ -\frac{1}{2}\Delta t^2 \mu^{p}{}^{[1]}_{1,2} - \Delta t \mu^{v}{}^{[1]}_{1,2} \end{bmatrix} + \sigma^{[1]}_0 \begin{bmatrix} 0 \\ 0 \\ v^{[1]}_{0,1} \\ v^{[1]}_{0,2} \\ 0 \\ 0 \end{bmatrix}$$
$$+ \nu^{[1]}_0 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u^{[1]}_{0,1} \\ u^{[1]}_{0,2} \end{bmatrix} + \lambda^{[1]}_{0,1} \begin{bmatrix} 4 - p^{[1]}_{0,1} \\ 1 - p^{[1]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[1]}_{0,2} \begin{bmatrix} 8 - p^{[1]}_{0,1} \\ 6 - p^{[1]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[1]}_{0,3} \begin{bmatrix} 9 - p^{[1]}_{0,1} \\ 0 - p^{[1]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[1]}_{0,4} \begin{bmatrix} 14 - p^{[1]}_{0,1} \\ 5 - p^{[1]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

$$\nabla_0 L^{[2]} = \begin{bmatrix} p^{[2]}_{0,1} - p^{[1]}_{0,1} \\ p^{[2]}_{0,2} - p^{[1]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \mu^{p}{}^{[2]}_{0,1} - \mu^{p}{}^{[2]}_{1,1} \\ \mu^{p}{}^{[2]}_{0,2} - \mu^{p}{}^{[2]}_{1,2} \\ \mu^{v}{}^{[2]}_{0,1} - \mu^{v}{}^{[2]}_{1,1} - \Delta t \mu^{p}{}^{[2]}_{1,1} \\ \mu^{v}{}^{[2]}_{0,2} - \mu^{v}{}^{[2]}_{1,2} - \Delta t \mu^{p}{}^{[2]}_{1,2} \\ -\frac{1}{2}\Delta t^2 \mu^{p}{}^{[2]}_{1,1} - \Delta t \mu^{v}{}^{[2]}_{1,1} \\ -\frac{1}{2}\Delta t^2 \mu^{p}{}^{[2]}_{1,2} - \Delta t \mu^{v}{}^{[2]}_{1,2} \end{bmatrix} + \sigma^{[2]}_0 \begin{bmatrix} 0 \\ 0 \\ v^{[2]}_{0,1} \\ v^{[2]}_{0,2} \\ 0 \\ 0 \end{bmatrix} + \nu^{[2]}_0 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u^{[2]}_{0,1} \\ u^{[2]}_{0,2} \end{bmatrix}$$
$$+ \lambda^{[2]}_{0,1} \begin{bmatrix} 4 - p^{[2]}_{0,1} \\ 1 - p^{[2]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[2]}_{0,2} \begin{bmatrix} 8 - p^{[2]}_{0,1} \\ 6 - p^{[2]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[2]}_{0,3} \begin{bmatrix} 9 - p^{[2]}_{0,1} \\ 0 - p^{[2]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[2]}_{0,4} \begin{bmatrix} 14 - p^{[2]}_{0,1} \\ 5 - p^{[2]}_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

$$\nabla_1 L^{[1]} = \omega_G \begin{bmatrix} p^{[1]}_{1,1} - 17 \\ p^{[1]}_{1,2} - 9 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \omega_P \begin{bmatrix} p^{[1]}_{1,1} - p^{[2]}_{1,1} \\ p^{[1]}_{1,2} - p^{[2]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \mu^{p\,[1]}_{1,1} - \mu^{p\,[1]}_{2,1} \\ \mu^{p\,[1]}_{1,2} - \mu^{p\,[1]}_{2,2} \\ \mu^{v\,[1]}_{1,1} - \mu^{v\,[1]}_{2,1} - \Delta t \mu^{p\,[1]}_{2,1} \\ \mu^{v\,[1]}_{1,2} - \mu^{v\,[1]}_{2,2} - \Delta t \mu^{p\,[1]}_{2,2} \\ -\frac{1}{2}\Delta t^2 \mu^{p\,[1]}_{2,1} - \Delta t \mu^{v\,[1]}_{2,1} \\ -\frac{1}{2}\Delta t^2 \mu^{p\,[1]}_{2,2} - \Delta t \mu^{v\,[1]}_{2,2} \end{bmatrix} + \sigma^{[1]}_1 \begin{bmatrix} 0 \\ 0 \\ v^{[1]}_{1,1} \\ v^{[1]}_{1,2} \\ 0 \\ 0 \end{bmatrix}$$
$$+ \nu^{[1]}_1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u^{[1]}_{1,1} \\ u^{[1]}_{1,2} \end{bmatrix} + \lambda^{[1]}_{1,1} \begin{bmatrix} 4 - p^{[1]}_{1,1} \\ 1 - p^{[1]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[1]}_{1,2} \begin{bmatrix} 8 - p^{[1]}_{1,1} \\ 6 - p^{[1]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[1]}_{1,3} \begin{bmatrix} 9 - p^{[1]}_{1,1} \\ 0 - p^{[1]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[1]}_{1,4} \begin{bmatrix} 14 - p^{[1]}_{1,1} \\ 5 - p^{[1]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

$$\nabla_1 L^{[2]} = \begin{bmatrix} p^{[2]}_{1,1} - p^{[1]}_{1,1} \\ p^{[2]}_{1,2} - p^{[1]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \mu^{p\,[2]}_{1,1} - \mu^{p\,[2]}_{2,1} \\ \mu^{p\,[2]}_{1,2} - \mu^{p\,[2]}_{2,2} \\ \mu^{v\,[2]}_{1,1} - \mu^{v\,[2]}_{2,1} - \Delta t \mu^{p\,[2]}_{2,1} \\ \mu^{v\,[2]}_{1,2} - \mu^{v\,[2]}_{2,2} - \Delta t \mu^{p\,[2]}_{2,2} \\ -\frac{1}{2}\Delta t^2 \mu^{p\,[2]}_{2,1} - \Delta t \mu^{v\,[2]}_{2,1} \\ -\frac{1}{2}\Delta t^2 \mu^{p\,[2]}_{2,2} - \Delta t \mu^{v\,[2]}_{2,2} \end{bmatrix} + \sigma^{[2]}_0 \begin{bmatrix} 0 \\ 0 \\ v^{[2]}_{1,1} \\ v^{[2]}_{1,2} \\ 0 \\ 0 \end{bmatrix} + \nu^{[2]}_0 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u^{[2]}_{1,1} \\ u^{[2]}_{1,2} \end{bmatrix}$$
$$+ \lambda^{[2]}_{1,1} \begin{bmatrix} 4 - p^{[2]}_{1,1} \\ 1 - p^{[2]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[2]}_{1,2} \begin{bmatrix} 8 - p^{[2]}_{1,1} \\ 6 - p^{[2]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[2]}_{1,3} \begin{bmatrix} 9 - p^{[2]}_{1,1} \\ 0 - p^{[2]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda^{[2]}_{1,4} \begin{bmatrix} 14 - p^{[2]}_{1,1} \\ 5 - p^{[2]}_{1,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

which are the first 24 rows of 612 Lagrangian equations ($6 \times 51 \times 2$) for this scenario with $N = 50$ time steps and $N_{\mathcal{I}} = 2$ players. The linear dynamics equality

constraints for each player $\mathcal{P}_i$, each associated with a free dual variable vector $\mu_0^{[i]}$, for $k = 0, 1$ are

$$h_0(x^{[i]}, u^{[i]}) = \begin{bmatrix} p_{0,1}^{[i]} \\ p_{0,2}^{[i]} \\ v_{0,1}^{[i]} \\ v_{0,1}^{[i]} \end{bmatrix} - \begin{bmatrix} p_{IC,1}^{[i]} \\ p_{IC,2}^{[i]} \\ v_{IC,1}^{[i]} \\ v_{IC,2}^{[i]} \end{bmatrix} = 0$$

$$h_1(x^{[i]}, u^{[i]}) = \begin{bmatrix} p_{1,1}^{[i]} \\ p_{1,2}^{[i]} \\ v_{1,1}^{[i]} \\ v_{1,2}^{[i]} \end{bmatrix} - \begin{bmatrix} p_{0,1}^{[i]} + \Delta t v_{0,1}^{[i]} \\ p_{0,2}^{[i]} + \Delta t v_{0,2}^{[i]} \\ v_{0,1}^{[i]} \\ v_{0,2}^{[i]} \end{bmatrix} - \begin{bmatrix} \frac{1}{2}\Delta t u_{0,1}^{[i]} \\ \frac{1}{2}\Delta t u_{0,2}^{[i]} \\ \Delta t u_{0,1}^{[i]} \\ \Delta t u_{0,2}^{[i]} \end{bmatrix} = 0$$

The velocity and acceleration bound complementarity equations for each player $\mathcal{P}_i$ are again

$$\begin{aligned} \frac{1}{2}\sigma_0^{[i]} \left[(v_{0,1}^{[i]})^2 + (v_{0,2}^{[i]})^2 - 5^2 \right] &= 0 \\ \frac{1}{2}\sigma_1^{[i]} \left[(v_{1,1}^{[i]})^2 + (v_{1,2}^{[i]})^2 - 5^2 \right] &= 0 \\ \frac{1}{2}\nu_0^{[i]} \left[(u_{0,1}^{[i]})^2 + (u_{0,2}^{[i]})^2 - 20^2 \right] &= 0 \\ \frac{1}{2}\nu_1^{[i]} \left[(u_{1,1}^{[i]})^2 + (u_{1,2}^{[i]})^2 - 20^2 \right] &= 0 \end{aligned}$$

Finally, the obstacle avoidance complementarity equations for each player $\mathcal{P}_i$ for the four obstacles are

$$\lambda_{0,1}^{[i]}\left[\frac{1}{2}\left(2^2-(4^2+1^2)-(p_{0,1}^{[i]})^2-(p_{0,2}^{[i]})^2\right)+4p_{0,1}^{[i]}+1p_{0,2}^{[i]}\right]=0$$
$$\lambda_{1,1}^{[i]}\left[\frac{1}{2}\left(2^2-(4^2+1^2)-(p_{1,1}^{[i]})^2-(p_{1,2}^{[i]})^2\right)+4p_{1,1}^{[i]}+1p_{1,2}^{[i]}\right]=0$$
$$\lambda_{0,2}^{[i]}\left[\frac{1}{2}\left(2^2-(8^2+6^2)-(p_{0,1}^{[i]})^2-(p_{0,2}^{[i]})^2\right)+8p_{0,1}^{[i]}+6p_{0,2}^{[i]}\right]=0$$
$$\lambda_{1,2}^{[i]}\left[\frac{1}{2}\left(2^2-(8^2+6^2)-(p_{1,1}^{[i]})^2-(p_{1,2}^{[i]})^2\right)+8p_{1,1}^{[i]}+6p_{1,2}^{[i]}\right]=0$$
$$\lambda_{0,3}^{[i]}\left[\frac{1}{2}\left(2^2-(9^2+0^2)-(p_{0,1}^{[i]})^2-(p_{0,2}^{[i]})^2\right)+9p_{0,1}^{[i]}+0p_{0,2}^{[i]}\right]=0$$
$$\lambda_{1,3}^{[i]}\left[\frac{1}{2}\left(2^2-(9^2+0^2)-(p_{1,1}^{[i]})^2-(p_{1,2}^{[i]})^2\right)+9p_{1,1}^{[i]}+0p_{1,2}^{[i]}\right]=0$$
$$\lambda_{0,4}^{[i]}\left[\frac{1}{2}\left(2^2-(14^2+5^2)-(p_{0,1}^{[i]})^2-(p_{0,2}^{[i]})^2\right)+14p_{0,1}^{[i]}+5p_{0,2}^{[i]}\right]=0$$
$$\lambda_{1,4}^{[i]}\left[\frac{1}{2}\left(2^2-(14^2+5^2)-(p_{1,1}^{[i]})^2-(p_{1,2}^{[i]})^2\right)+14p_{1,1}^{[i]}+5p_{1,2}^{[i]}\right]=0$$

The complete system of equations in this scenario includes 1614 equations and 1614 variables (both primal and Lagrangian multipliers). The equilibrium trajectory solution to such a pursuit-evasion game is depicted in Figure 5.1, where the attacker reaches its target at time step $k = 40$ (i.e., time $t_k = 4$ s). Figure 5.2 shows the evolution of each player's payoff over time.

The attacker is represented by the red diamond starting at $(0, 0)$, while the blue square represents the defender starting at $(0, 10)$, with the target represented by a green triangle at $(17, 9)$, and circular obstacles throughout the work space. If the defender fails to intercept the attacker directly, its objective function levels off

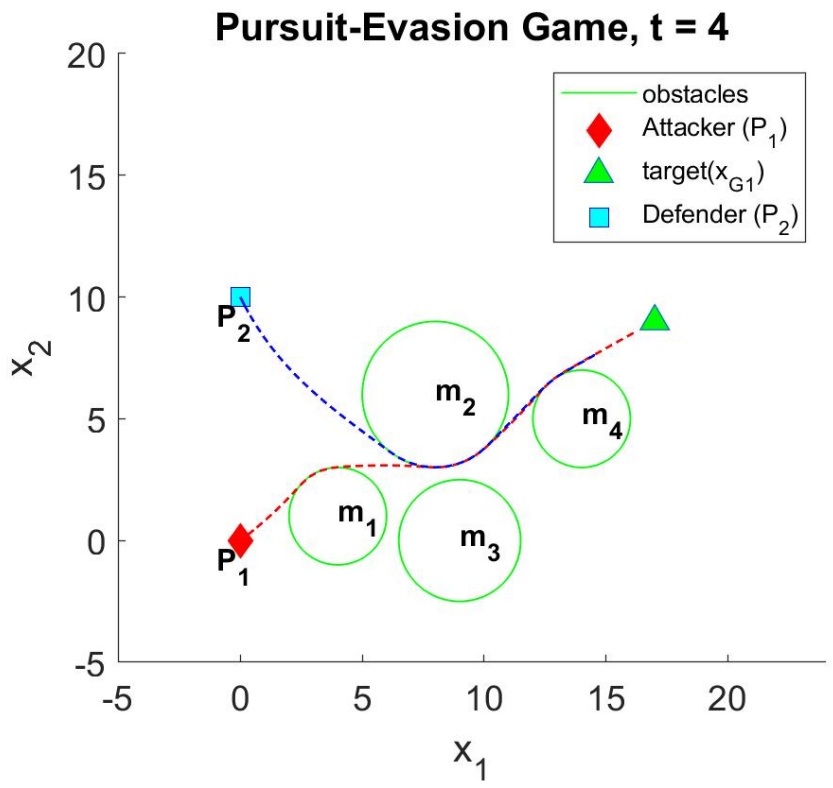


Figure 3.5: Pursuit-Evasion Game Trajectory with one Defenders, one Attacker

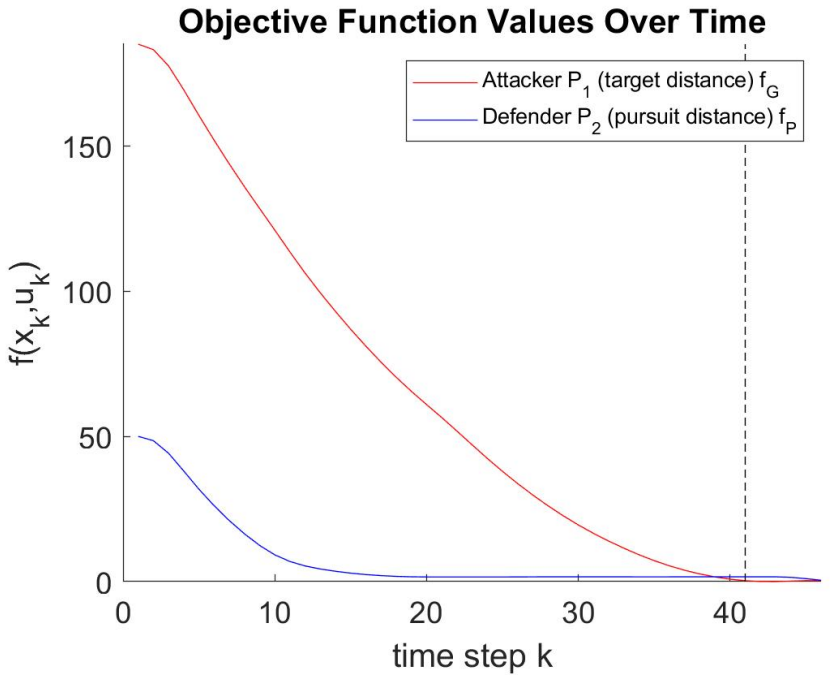


Figure 3.6: Objective Values for one Defender, one Attacker PEG

when the pursuer starts to follow the same path as the attacker since all vehicles have the same max speed v_{max} in this scenario.

3.2.4 Nonlinear Program Post-Check

An important validation step in accepting the MCP solution is performing a NLP post-check. The concept is to simply solve the same problem for each player while holding the other player's actions stationary as solved in the MCP. Thus, for $\mathcal{P}_1$'s optimization problem, the state vector $x^{[2]}$ and control vector $u^{[2]}$ for $\mathcal{P}_2$ become fixed, and vice-versa. The results of this post-check are shown in Table 3.3. The

Table 3.3: Two-Player PEG Results via MCP

Cost Value	MCP Solution	NLP Post-check
F_G	2851.06	2851.06
F_P	371.53	371.53
Result	Goal	Goal
t_F	4	4

columns match exactly, as they should, since the solution generated by the MCP seems to satisfy KKT optimality conditions, implying at least a local optimum to the NLP for each player, provided that the actions of the other player don't change.

Chapter 4

Rolling-Horizon Robust Equilibrium

4.1 Definition of Rolling-Horizon Foresight

An alternative approach to solving the pursuit-evasion game as one large, deterministic equilibrium problem is to use a rolling horizon, in which each vehicle only has limited local information about the state space. The benefits to such an approach are two-fold; in that computational costs per vehicle $\mathcal{P}_i$ are subject only to the information it has available at the start of the planning horizon rather than the entire workspace, and that our model more closely resembles actual operations in which no agent is omniscient. Recall that each vehicle acts independently, and so we continue to use the terms "player" and "vehicle" interchangeably.

Therefore, a master problem MP can be split into sub-problems SP_1,..., SP_h,..., SP_H, each of the form presented in Chapter 2. We split the time series K into subsets of time indices defined as $K_h = \{k_0(h), ..., k_N(h)\} \in K$, where the initial time step $k_0(h) = (h-1)N$ and final step $k_N(h) = hN$. This assumes an equal number of time steps per horizon, and ensures that each planning horizon h

includes exactly $N+1$ collocation points, i.e.,

$$\begin{aligned} h=1: &\quad K_1=\{0,...,N\} \\ h=2: &\quad K_2=\{N,...,2N\} \\ &\vdots \\ h=H: &\quad K_H=\{(H-1)N,...,HN\} \end{aligned}$$

where H is a finite maximum number of planning horizons to consider, and N is the number of time steps in each horizon. Since each sub-problem is solved independently, it is necessary to first define the sub-path trajectory vector $x_{(h)}^{[i]}$ and sub-path control vector $u_{(h)}^{[i]}$ (where i is the vehicle index for $\mathcal{P}_i$) for each time horizon h:

$$\begin{aligned} x_{(h)}^{[i]} &\triangleq (x_{k0}^{[i]T},...,x_{k}^{[i]T},...,x_{kN}^{[i]T})^T \in \mathcal{X} \subseteq \mathbb{R}^{4(N+1)}, \\ u_{(h)}^{[i]} &\triangleq (u_{k0}^{[i]T},...,u_{k}^{[i]T},...,u_{kN-1}^{[i]T})^T \in \mathcal{U} \subseteq \mathbb{R}^{2N}, \end{aligned}$$

and now, we can define $\mathcal{P}_i$'s strategy for horizon h, $s_h^{[i]}=(x_{(h)}^{[i]},u_{(h)}^{[i]})$, as well as the joint set of strategies $s_h=(s_h^{[i]},\forall i\in\mathcal{P})\in\mathcal{S}_{(h)}\subseteq\mathcal{S}$. For bookkeeping purposes, we also define a global trajectory $X^{[i]}$ and global control vector $U^{[i]}$ which denote the

combined paths and control decisions for player $\mathcal{P}_i$ across all horizons

$$X^{[i]} \triangleq (x_{(1)}^{[i]T}, x_{(2)}^{[i]T}, ..., x_{(h)}^{[i]T}, ..., x_{(H)}^{[i]T})^T,$$

$$U^{[i]} \triangleq (u_{(1)}^{[i]T}, u_{(2)}^{[i]T}, ..., u_{(h)}^{[i]T}, ..., u_{(H)}^{[i]T})^T.$$

The transitions between planning horizons serve as continuity points, such that the time index k_N and state vector $x_{k_N}^{[i]}$ at the end of one planning horizon are saved as the initial conditions k_0 and state vector $x_{k_0}^{[i]}$ in the next planning horizon (e.g., $x_{k_N}^{[i]}|_h = x_{k_0}^{[i]}|_{h+1}$). This concept is depicted visually in Figure 4.1, where $N = 10$ and $H = 5$.

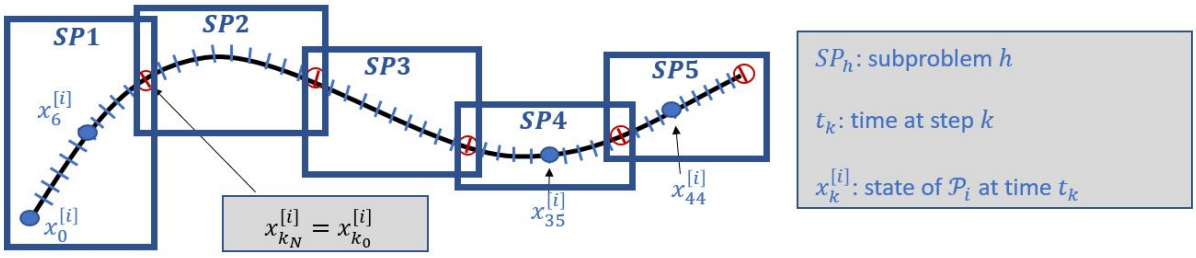


Figure 4.1: Rolling-Horizon Sub-Problem Concept with Continuity Points

Recall that the pursuit-evasion game presented in this thesis includes a distinct set of attacker UUVs $\mathcal{P}_\mathcal{A}$ and another distinct set of defender UUVs $\mathcal{P}_\mathcal{D}$; where in the current chapter $\mathcal{P}_\mathcal{A} = \{\mathcal{P}_1\}$ and $\mathcal{P}_\mathcal{D} = \{\mathcal{P}_2\}$. Also, the notation $\mathcal{P}_i$, when used, refers generally to any player in the system. For the purpose of this thesis, we consider a scenario in which the attacker vehicle $\mathcal{P}_1$ has global information about its own states $x_k^{[1]}$ and target position p_G, while the defender vehicle $\mathcal{P}_2$ has global information only about its own states $x_k^{[2]}$.

However, both vehicles only have "local" information about obstacles and their adversaries. Here "local" is interpreted in terms of Euclidean distance from the vehicle $\mathcal{P}_i$'s position at the start of each rolling horizon, $p_{k_0}^{[i]}$. This leads us to define a detection radius $r_\delta^{[i]}$ for each vehicle $\mathcal{P}_i$, inside of which objects become local information, and outside of which objects can be completely unknown or nominally mapped as robust obstacles, which is presented in Section 4.2.1.

Note that with a rolling-horizon foresight, a vehicle $\mathcal{P}_i$ is prone to finding a sub-path $x_h^{[i]}$ that is locally optimal in the current horizon h, but that may result in a globally sub-optimal path $X^{[i]}$ and control vector $U^{[i]}$ due to unforeseen obstacles and adversaries outside its detection range.

4.1.1 Foresight and Safety

We assume in this case that the detection radius $r_\delta^{[i]}$ is always greater than the travel range r_h in each time horizon h, where r_h is an implicit parameter defined by the maximum velocity v_{max} and time horizon t_N, such that $r_h = t_N v_{max}$. The travel range is the maximum distance the vehicle can travel during a single planning horizon h if it is already traveling at max speed (i.e., $\|v_{k_0}\| = v_{max}$), and does not change direction. To demonstrate this, we introduce a displacement function

$$d_h^{[i]}(x_{(h)}^{[i]}, u_{(h)}^{[i]}) = \sum_{k=k_0}^{k_N-1} \|p_{k+1}^{[i]} - p_k^{[i]}\|, \quad \forall \ h = 1, ..., H \tag{4.1}$$

which determines a linear approximation of the physical sub-path length of $x_{(h)}^{[i]}$, or distance traveled, from the initial position $x_{k_0}^{[i]}$ to the final position $x_{k_N}^{[i]}$. These

assumptions are also stated below in Assumption **4.1**.

Assumption 4.1 (Sufficient Foresight) *All vehicles $\mathcal{P}_i$ have a sufficient detection radius $r_\delta^{[i]}$ strictly greater than the travel range $r_h^{[i]}$, which, in turn, is always equal or greater than the vehicle's displacement $d_h^{[i]}$.*

$$r_\delta^{[i]} > r_h^{[i]} = t_N v_{max}^{[i]} \geq d_h^{[i]}.$$

which keeps the vehicle $\mathcal{P}_i$ from traveling beyond a range that it can detect obstacles or other vehicles in any single planning horizon h. The relationship between r_h and $r_\delta^{[i]}$ is shown in Figure 4.2.

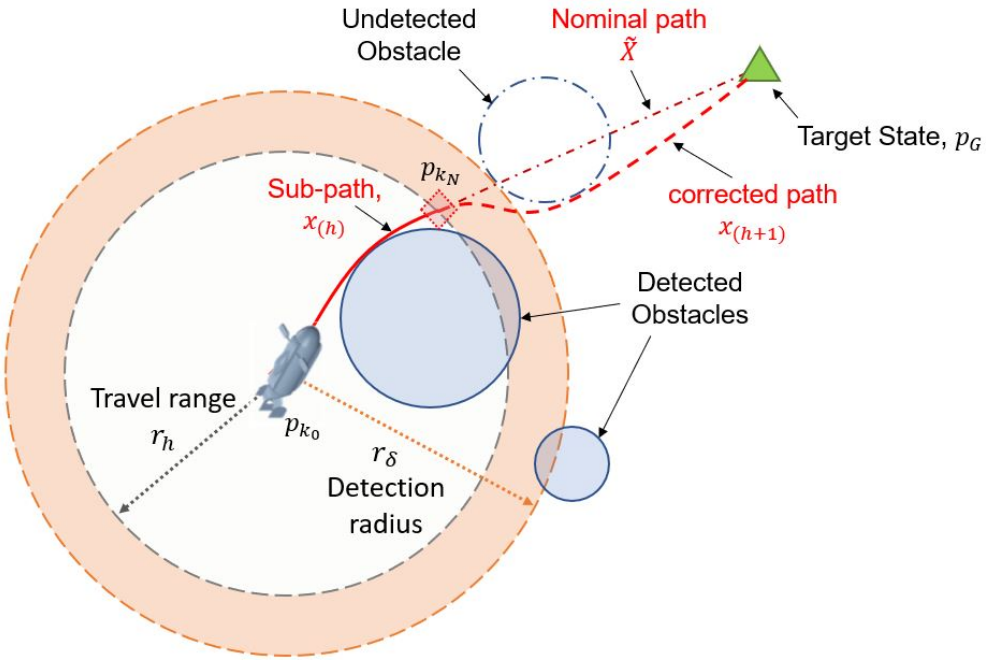


Figure 4.2: Rolling-Horizon Foresight with Obstacle Detection

The space between the detection radius (orange circle) and travel range (gray circle) can be considered a safety buffer. The larger this region, the better foresight the vehicle has beyond its physical travel capabilities. Note in the figure that the

sub-path $x_{(h)}$ does not consider the undetected obstacle outside r_δ, and so the vehicle has to adjust course in the next sub-path $x_{(h+1)}$.

4.1.2 Obstacle Detection

Given the definition for local information and detection above, we can define an obstacle detection function

$$\delta_{(m)}^{[i]} \triangleq \mathbf{1}\{\|p_{k_0}^{[i]} - c_m\| \leq r_m + r_\delta^{[i]}\}, \tag{4.2}$$

where $\mathbf{1}\{\cdot\}$ is an indicator function that "activates" the conditional obstacle avoidance constraint for vehicle $\mathcal{P}_i$ and obstacle $\mathcal{O}_{(m)}$ only if that obstacle boundary falls within the detection radius (i.e., $\delta_{(m)}^{[i]} = 1$). Thus, obstacles described in Section 2.2.4 become visible to player $\mathcal{P}_i$ via

$$\mathcal{O}_{(m)}^{[i]} \triangleq \{\delta_{(m)}^{[i]} p \in \mathcal{W} : ||p - c_m(k)|| \leq r_m\}, \tag{4.3}$$

which generates conditional obstacle avoidance constraint, $g_{km}^{obs}(x_k^{[i]})$, $\forall k \in K_h, m \in M^{[i]}$ in $\mathcal{P}_i$'s optimization in the same form as (3.1f) only if $\delta_{(m)}^{[i]} = 1$. The diagram shown in Figure 4.3 describes how this is implemented. Note that the nearest point on the surface of the first obstacle (top) is outside the detection area (i.e., beyond the detection radius r_δ from $\mathcal{P}_i$'s initial position p_{k_0}), so that obstacle $\mathcal{O}_{(1)}$ is undetected, while $\mathcal{O}_{(2)}$ (bottom) is detected in the current horizon.

The conditional constraint is included/excluded prior to the problem reach-

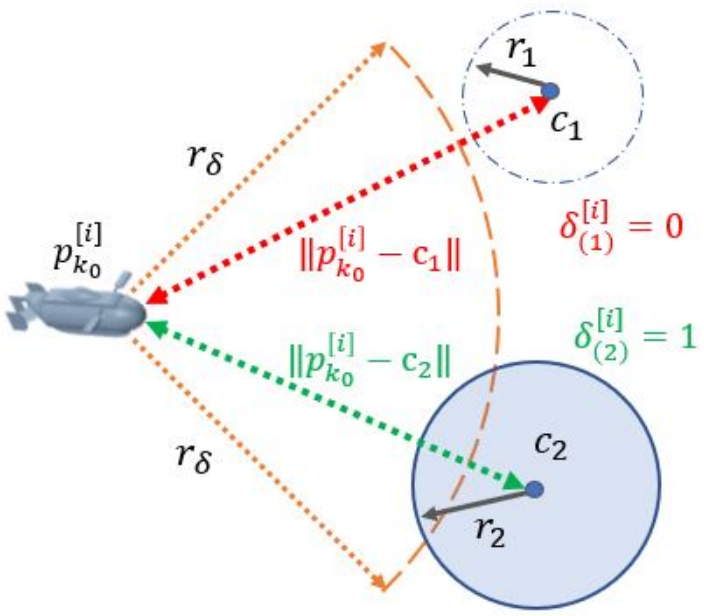


Figure 4.3: Obstacle Detection Concept

ing the GAMS PATH solver, so that discrete binary variables can be avoided in the MCP. Also, this algorithm implicitly induces a sort of "memorylessness" property in our system, since vehicles do not store obstacle locations after they have passed them. Although current vehicle platforms are capable of mapping and storing this type of information, doing so only induces unnecessary obstacle-avoidance constraints in our model since vehicles tend not to backtrack in a shortest path to a fixed target.

4.1.3 Vehicle Detection

Similarly, while obstacle detection impacts the number of constraints considered, vehicle detection affects the objective function, in that the pursuit function $f_{Pk}^{[i]}$ should only be an active sub-objective for player $\mathcal{P}_i$ in horizon h if player $\mathcal{P}_j$'s

position ($i \neq j$) is within $r_\delta^{[i]}$ at time step k_0. Formally, this is stated as:

$$\gamma_j^{[i]} = \mathbf{1}\{\|p_{k_0}^{[i]} - p_{k_0}^{[j]}\| \leq r_\delta^{[i]}\}, \quad \forall i, j \in \{1, ..., |\mathcal{P}|\}, i \neq j, \tag{4.4}$$

which conditionally includes the pursuit function from (3.7a) in $\mathcal{P}_i$'s objective, such that

$$f_P^{[i]} = \gamma_j^{[i]} \frac{1}{2} \|p_k^{[j]} - p_k^{[i]}\|^2 \tag{4.5}$$

Since each vehicle presumably has its own unique detection radius, $r_\delta^{[i]}$ based on vehicle-platform, it is possible that $\mathcal{P}_i$ detects $\mathcal{P}_j$ before $\mathcal{P}_j$ detects $\mathcal{P}_i$ if $r_\delta^{[i]} < r_\delta^{[j]}$. In this thesis, we assume all vehicles have the same detection radius $r_\delta^{[1]} = r_\delta^{[2]} = r_\delta$.

Assumption 4.2 (Detection Equity) *All vehicles have the same detection radius r_δ, and thus $\gamma_j^{[i]} = \gamma_i^{[j]}$ $\forall i \neq j$, so no advantage of information exists.*

4.1.4 Rolling-Horizon Equilibrium Problem

By implementing vehicle and obstacle detection into the attacker vehicle $\mathcal{P}_1$'s optimization problem (3.8a), we arrive at a new formulation in which $\mathcal{P}_1$ wants to

minimize the cost functional $J_h^{[1]}(s_h)$ for every planning horizon $h \in \{1, ..., H\}$:

$$\min_{u_{(h)}^{[1]}} \; J_h^{[1]}(s_h) \triangleq \frac{1}{2} \sum_{k=0}^{k_N} \left[\omega_G \|p_G - p_k^{[1]}\|^2 - \omega_P \gamma_2^{[1]} \|p_k^{[2]} - p_k^{[1]}\|^2 \right] \tag{4.6a}$$

s.t.

$$x_0^{[1]} = x_{IC}^{[1]} \tag{4.6b}$$

$$x_{k+1}^{[1]} = A x_k^{[1]} + B u_k^{[1]} \quad \forall \; k = k_0, ..., k_N - 1 \tag{4.6c}$$

$$\frac{1}{2} \left\| v_k^{[1]} \right\|^2 \leq \frac{1}{2} v_{max}^2 \quad \forall \; k \in K_h \tag{4.6d}$$

$$\frac{1}{2} \|u_k^{[1]}\|^2 \leq \frac{1}{2} a_{max}^2 \quad \forall \; k = k_0, ..., k_N - 1 \tag{4.6e}$$

$$\frac{1}{2} \left\| p_k^{[1]} - c_m \right\|^2 \geq \frac{1}{2} r_m^2 \quad \forall \; k \in K_h, m \in M^{[1]} \tag{4.6f}$$

where $M^{[1]}$ is the set of obstacles currently detected by player $\mathcal{P}_1$,

$$M^{[1]} \triangleq \{m : \delta_{(m)}^{[1]} = 1\} \subseteq M,$$

and is updated each planning horizon.

Likewise, the defender $\mathcal{P}_2$ wants to minimize the cost functional $J_h^{[2]}(s_h)$ in each horizon $h \in \{1, ..., H\}$, which includes only the pursuit functions $f_{Pk}^{[2]}$, resulting

in:

$$\min_{u_{(h)}^{[2]}} \quad J_h^{[2]}(s_h) \triangleq \gamma_1^{[2]} \frac{1}{2} \sum_{k=k_0}^{k_N} \|p_k^{[2]} - p_k^{[1]}\|^2 \tag{4.7a}$$

s.t.

$$x_0^{[2]} = x_{IC}^{[2]} \tag{4.7b}$$

$$x_{k+1}^{[2]} = Ax_k^{[2]} + Bu_k^{[2]} \quad \forall \ k = k_0, ..., k_N - 1 \tag{4.7c}$$

$$\frac{1}{2}\left\|v_k^{[2]}\right\|^2 \leq \frac{1}{2} v_{max}^2 \quad \forall \ k \in K_h \tag{4.7d}$$

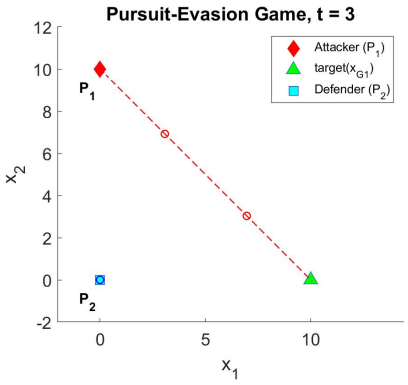
$$\frac{1}{2}\|u_k^{[2]}\|^2 \leq \frac{1}{2} a_{max}^2 \quad \forall \ k = 0, ..., k_N - 1 \tag{4.7e}$$

$$\frac{1}{2}\left\|p_k^{[2]} - c_m\right\|^2 \geq \frac{1}{2} r_m^2 \quad \forall \ k \in K_h, m \in M^{[2]} \tag{4.7f}$$

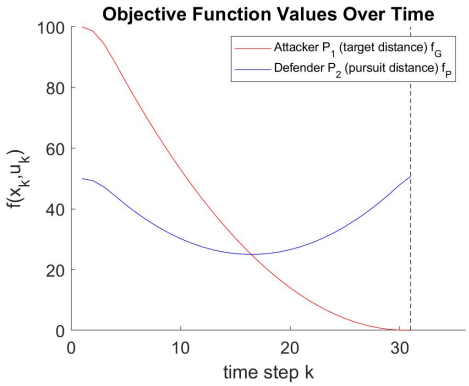
where $M^{[2]} \subseteq M$ is the set of obstacles currently detected by player $\mathcal{P}_2$. Note that if $\gamma_2^{[1]} = \gamma_1^{[2]} = 0$, then player $\mathcal{P}_1$ minimizes $J_h^{[1]}$ in (4.6) by simply moving towards its target p_G and player $\mathcal{P}_2$ minimizes $J_h^{[2]} = 0$ in (4.7) by remaining stationary. This is visually demonstrated in Figure 4.4 in which $r_\delta < \|p_{k_0}^{[2]} - p_{k_0}^{[1]}\|$, so $\mathcal{P}_1$ takes a direct route from its initial position $(0, 10)$ to its target $(10, 0)$ and $\mathcal{P}_2$ remains at $(0, 0)$. However, if the detection radius is increased to $r_\delta = 10$ then the defender $\mathcal{P}_2$ pursues $\mathcal{P}_1$, while the attacker $\mathcal{P}_1$ attempts to evade, resulting in capture just before reaching the target, shown in Figure 4.5.

4.2 Incomplete Information

In underwater path planning, agents make decisions with incomplete information. Thus, there are many extensions that could be further explored with the



(a) Trajectories for 2-Player PEG

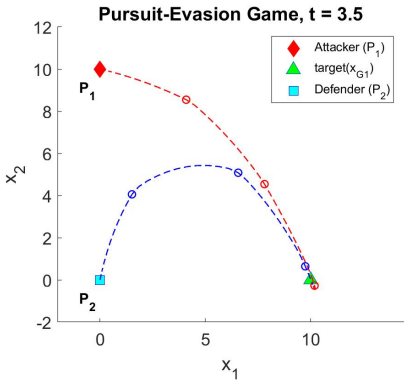


(b) Cost Functions over Time

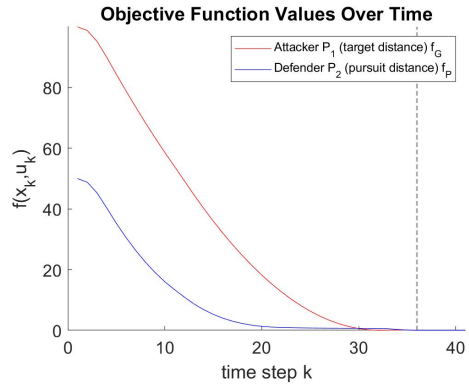
Figure 4.4: No Pursuit when $r_\delta = 5 < \|p_{k0}^{[2]} - p_{k0}^{[1]}\|$

rolling-horizon approach. For instance, a vehicle could only have partial information about the size and location of obstacles, its target location and even its own position due to the nature of its environment, where the GPS signal is either reduced or unavailable [34]. For now, we maintain the following assumption:

Assumption 4.3 (Perfect Self-Awareness) *Player $\mathcal{P}_i$ retains perfect current knowledge of its own state $x_k^{[i]}$ at time step k, but only possesses nominal information, $\hat{c}_m^{[i]}$ and $\hat{r}_m^{[i]}$, about obstacles observed at the start of each horizon.*



(a) Trajectories for 2-Player PEG



(b) Cost Functions over Time

Figure 4.5: Defender $\mathcal{P}_2$ pursues $\mathcal{P}_1$ when $r_\delta = 10 \geq \|p_{k0}^{[2]} - p_{k0}^{[1]}\|$

This assumption supposes that vehicles know their own position and velocity, but may only possess partial information about obstacles (exogenous uncertainty) and/or the effect of internal control on their vehicle dynamics (endogenous).

4.2.1 Interval Robustness

We now introduce the concept of robustness to our equilibrium model, which can be interpreted to have a number of meanings. We first introduce a sensor confidence parameter $\rho^{[i]} \in [0, 1]$ that defines the level of certainty vehicle $\mathcal{P}_i$ has about its environment. So when $\rho^{[i]} = 0$ the vehicle has no certainty about the actual size and location of obstacles that it detects, while when $\rho^{[i]} = 1$ it can navigate with deterministic certainty. We then observe how the circular obstacles $\mathcal{O}$ can be modified in the state space to efficiently account for interval robustness [22] in their location and size parameters. This means that the centers and radii of obstacles can fall within a permissible range around a nominal value. Then we incorporate this concept with a rolling-horizon foresight.

4.2.2 Robust Obstacles

Robust obstacles are approached in a worst-case representation, where obstacle attributes are stochastic, in that player $\mathcal{P}_i$ detects a nominal center $\hat{c}_m^{[i]}$ and nominal radii $\hat{r}_m^{[i]}$ that may vary from the actual values for each obstacle. The observed nominal center $\hat{c}_m^{[i]}$ and radius $\hat{r}_m^{[i]}$ reported by the vehicle sensor are known to vehicle $\mathcal{P}_i$, but the actual center realization c_m may vary from the observed center within

$\pm\Delta c_{max}$, and the actual obstacle radius r_m may also vary by $\pm\Delta r_{max}$, which are defined:

$$\Delta c_{max}^{[i]} \triangleq (1-\rho^{[i]})\frac{\left(\|c_m - p_{k_0}^{[i]}\| - r_m\right)^2}{2r_\delta^{[i]}}, \tag{4.8a}$$

$$\Delta r_{max}^{[i]} \triangleq (1-\rho^{[i]})\frac{r_m}{\|c_m - p_{k_0}^{[i]}\|} \tag{4.8b}$$

where the numerator in (4.8a) reflects the distance from the vehicle's sensor to the surface of the obstacle. Dividing by twice the vehicle's sensor range reduces the uncertainty within the detection range. This function is closely related to the obstacle avoidance constraint (3.7f). In (4.8b) the uncertainty is inversely proportional to its proximity to the center c_m, meaning that , for obstacles with large radii r_m, the closer it is to the obstacle, the wider the radius might vary. This gives the players the definitions of their nominal obstacles

$$\hat{c}_m^{[i]} \triangleq= c_m + \Delta c^{[i]} = \begin{bmatrix} c_{m,1} + \Delta c_1^{[i]} \\ c_{m,2} + \Delta c_2^{[i]} \end{bmatrix}, \quad |\Delta c_1^{[i]}|, |\Delta c_2^{[i]}| \leq \Delta c_{max}^{[i]} \tag{4.8c}$$

$$\hat{r}_m^{[i]} \triangleq r_m + \Delta r_m^{[i]}, \quad |\Delta r_m^{[i]}| \leq \Delta r_{max}^{[i]}, \tag{4.8d}$$

which vary depending on each player's confidence parameter $\rho^{[i]}$ and their respective distances from each obstacle. The uncertainty can be mitigated using a finite number of scenarios to capture an infinite number of scenarios within these robustness bounds. This is achieved by representing the obstacles as shown in Figure 4.6, where

a larger effective radius r'_m is generated around the outermost obstacle realizations given the definitions of $\hat{c}_m$ and $\hat{r}_m$

$$r'^{[i]}_m \triangleq \hat{r}^{[i]}_m + \max\{\sqrt{2}\Delta c^{[i]}_{max}, \Delta r^{[i]}_{max}\}, \tag{4.9}$$

where $\hat{r}^{[i]}_m$ is the observed nominal radius, and the second term represents the maximum change if either: a) both center coordinates shift maximally, $\Delta c_1 = \Delta c_2 = \Delta c_{max}$ (note: the coefficient $\sqrt{2}$ is the hypotenuse of a right triangle with legs of length 1), or b) the radius shifts maximally, $\Delta r = \Delta r_{max}$. By replacing the infinite number of potential realizations of obstacles within an acceptable range, this technique effectively and efficiently converts a stochastic problem into a deterministic equivalent problem. This provides a robust path for player $\mathcal{P}_i$ via

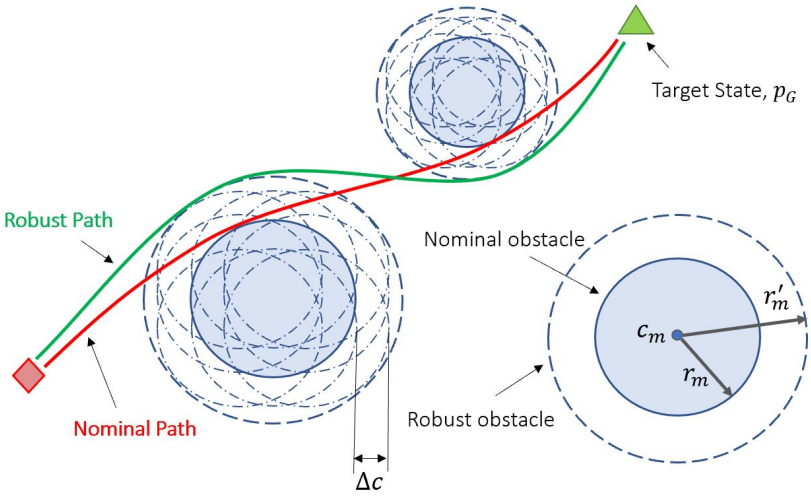


Figure 4.6: Robust Obstacle Avoidance

$$g^{rob}_{km}(x^{[i]}_k) \triangleq \frac{1}{2}\left(r'^{[i]2}_m - \left\|p^{[i]}_k - \hat{c}^{[i]}_m\right\|^2\right) \leq 0 \quad \forall\ k \in K_h, m \in M^{[i]} \quad (\lambda^{[i]}_{km}), \tag{4.10}$$

given that obstacles are sparsely distributed in the state-space, necessitating the following assumption.

Assumption 4.4 (Sparse Obstacles) *Separation between obstacles is greater than each of their respective radii , and no set of obstacles are configured such that more than half of any vehicle's detection area is restricted. Thus, no two obstacles overlap and are sparsely distributed.*

However, since this is a highly conservative approach, the paths might become sub-optimal relative to the path that would be selected given only the nominal obstacles if uncertainty bounds stretch potential obstacles outside the radius of the nominal obstacle. Also, modeling obstacles in this manner introduces potential for artificial infeasibility, meaning that even if a feasible path exists through actual obstacle realizations, the vehicle may fail to find it due to overlapping conservative estimates.

4.3 Rolling-Horizon Robust Equilibrium

Combining the two concepts of rolling-horizon foresight and robust obstacles produces a model that can represent a multi-player system with incomplete information and learning, which is of value in the verification and validation of UUV mission-planning algorithms and software. The KKT conditions for the rolling-

horizon robust game for each horizon h now become:

$\forall$ players $\mathcal{P}_i \in \mathcal{P}$, $\forall$ time steps $k \in K_h$:

$$\nabla_k L_h^{[i]}(s) \triangleq \nabla_k J_h^{[i]}(s) + \nabla h_k(x_{(h)}^{[i]}, u_{(h)}^{[i]})\mu_k^{[i]} + \sigma_k^{[i]} \nabla g_k^{vel}(x_k^{[i]}) \tag{4.11a}$$

$$+ \nu_k^{[i]} \nabla g_k^{acc}(u_k^{[i]}) + \sum_{m \in M} \lambda_{km}^{[i]} \nabla g_{km}^{rob}(x_k^{[i]}) = 0 \ , \ x_k, u_k \text{ free}$$

$$h_k(x_{(h)}^{[i]}, u_{(h)}^{[i]}) = 0 \ , \ \mu_k^{[i]} \text{ free} \quad . \tag{4.11b}$$

$$g_k^{vel}(x_k^{[i]}) \leq 0 \ \perp \ \sigma_k^{[i]} \geq 0 \tag{4.11c}$$

$$g_k^{acc}(u_k^{[i]}) \leq 0 \ \perp \ \nu_k^{[i]} \geq 0 \tag{4.11d}$$

$$g_{km}^{rob}(x_k^{[i]}) \leq 0 \ \perp \ \lambda_{km}^{[i]} \geq 0 \quad \forall \ m \in M^{[i]} \tag{4.11e}$$

where the cost functional $J_h^{[i]}(s)$, dynamics $h_k(x_{(h)}^{[i]}, u_{(h)}^{[i]})$ and bound constraints $g_k^{vel}(x_k^{[i]})$, $g_k^{acc}(u_k^{[i]})$ are defined by (4.6a)-(4.6e) for $\mathcal{P}_1$ and by (4.7a)-(4.7e) for $\mathcal{P}_2$, while robust obstacle avoidance constraints $g_{km}^{rob}(x_k^{[i]})$ for each player $\mathcal{P}_i$ are defined by (4.10). The procedure for finding the rolling-horizon robust equilibrium solution for the two-player pursuit evasion game is summarized in Algorithm 2

4.3.1 Learning via Objective Weighting for Attacker $\mathcal{P}_1$

Learning behavior can be challenging to understand, let alone model in a mathematical construct. However, using a rolling-horizon approach, we can implement an iterative learning mechanism. This is done for the attacker $\mathcal{P}_1$ by tuning the objective weighting parameters for the goal function ω_G and pursuit function ω_P based on proximity and relative position to the target and defender $\mathcal{P}_2$. If, for

Algorithm 2 Rolling-Horizon Robust Equilibrium

Input: $\mathcal{P}_1$ data ($x_{IC}^{[1]}$,p_G,v_{max},a_{max},ω_G), $\mathcal{P}_2$ data ($x_{IC}^{[2]}$,v_{max},a_{max}), obstacle data $\mathcal{O}$, confidence level ρ, detection range r_δ, dynamics matrices A and B, step size Δt, horizon length T_h, max horizons H

Output: $X^{[i]}$, $U^{[i]}$ $\forall i = 1, 2$, $result$, residual

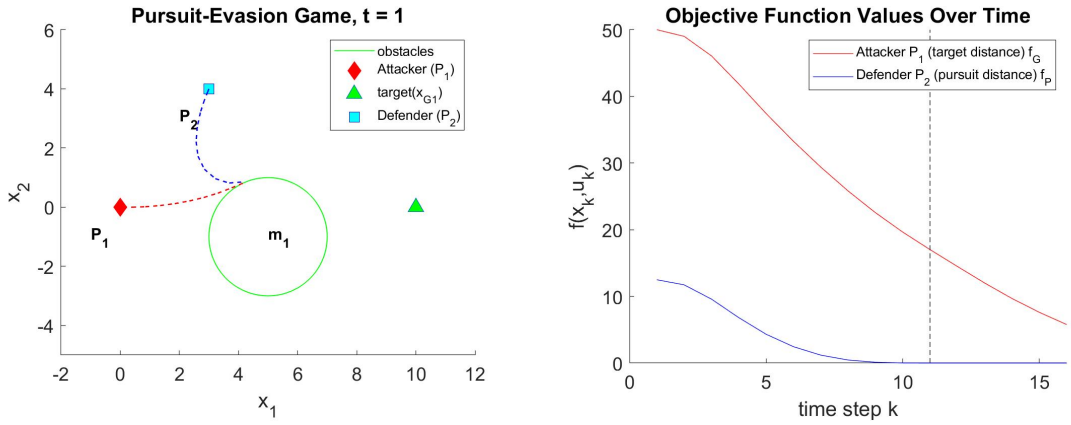
for $h \leftarrow 1$ to H **do**
  Find KKT solution via (4.11) without obstacle constraints g_k^{rob} (4.11e).
  $(x_{(h)}^{[i]}, u_{(h)}^{[i]}) \leftarrow (\bar{x}_{(h)}^{[i]}, \bar{u}_{(h)}^{[i]})$ $\forall i = 1, 2$ ▷ set initial guess
  $\mathcal{O}_{(m)}^{[i]} \leftarrow (\hat{c}_m^{[i]T}, r^{[i]\prime}{}_m)$ via (4.2),(4.8c),(4.9) ▷ update robust obstacles
  Find full KKT solution via (4.11).
  for $k \leftarrow k_0$ to k_N **do** ▷ check termination criteria
    if $\|p_k^{[1]} - p_G\| < \varepsilon_G$ **then**
      $result = 1, k_N \leftarrow k$ ▷ attacker reached target
      return
    end if
    if $\|p_k^{[2]} - p_k^{[1]}\| < \varepsilon_C$ **then**
      $result = 2, k_N \leftarrow k$ ▷ defender intercepted attacker
      return
    end if
  end for
end for

instance, $\mathcal{P}_1$ is positioned between the target p_G and defender $\mathcal{P}_2$ at the start of a planning horizon, and no other defenders are present, then the most appropriate objective weights would be $\omega_G = 1, \omega_P = 0$, or to maximize the goal function f_G. Recall $\mathcal{P}_1$'s explicit objective to reach the target p_G as quickly as possible without being intercepted by $\mathcal{P}_2$, so this adjustment makes sense as long as no obstacles are present between $\mathcal{P}_1$'s starting position $p_{k_0}^{[1]}$ and its target p_G.

While the full goal function weight works when $\mathcal{P}_1$ has a clear path to p_G, there are other scenarios in which the pursuit function outweighs the goal function ($\omega_P > \omega_G$) provides better opportunity to $\mathcal{P}_1$. Consider the game with the following parameters which generates the scenario pictured in Figure 4.7. Note that $\mathcal{P}_2$ captures $\mathcal{P}_1$ at $t = 0.9$ when $\mathcal{P}_1$'s objective is equally weighted with $\omega_G = \omega_P = 0.5$.

Table 4.1: Learning Game Parameters

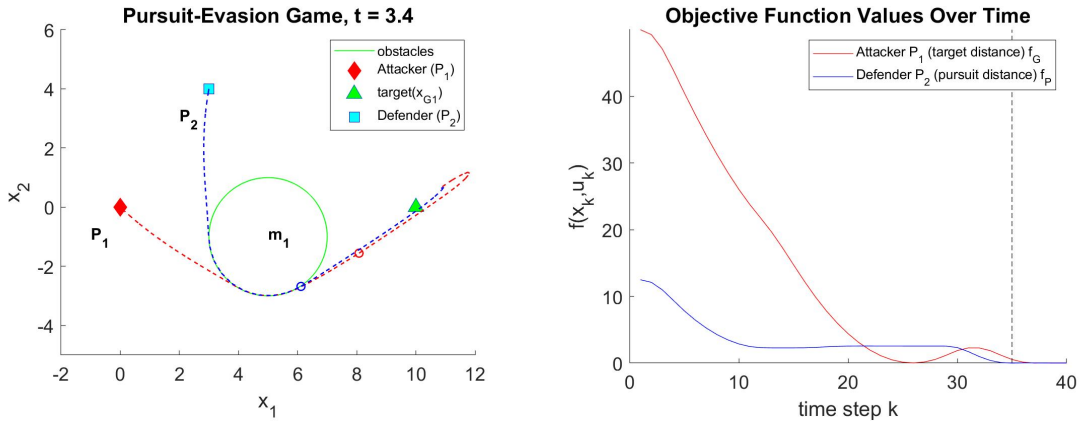
Parameter	Value	Units	Description
t_0	0	h	start time
Δt	0.1	h	time step
T_h	2	h	horizon length
$x_{IC}^{[1]}$	$(0,0,0,0)^T$	km	$\mathcal{P}_1$ initial state
$x_{IC}^{[2]}$	$(3,4,0,0)^T$	km	$\mathcal{P}_2$ initial state
x_G	$(10,0,0,0)^T$	km	target point
v_{max}	5	km/h	max speed
a_{max}	20	km/h^2	max acceleration
$\mathcal{O}_{(1)}$	$(5,-1,2)^T$	km	obstacle data
ω_G	0.5		goal obj weight



(a) Trajectories for 2-Player PEG (b) Cost Functions over Time

Figure 4.7: Balanced Objective, $\omega_G = \omega_P = 0.5$. Capture at $t = 0.9$

In the same scenario, since $\mathcal{P}_1$ is closer to $\mathcal{P}_2$ than the target p_G, if its objective is heavily weighted towards evasion (i.e., maximizing the pursuit function f_P), it puts itself in a position to reach its goal state p_G during the first of two sub-paths, as shown in in Figure 4.8. However, maintaining these objective weights in the second horizon starting at the terminal point denoted by the small red circle between the obstacle and the target in Figure 4.8a , $\mathcal{P}_1$ overshoots the target p_G and is instead captured at $t = 3.4$. This shows why the attacker might want to modify its objective



(a) Trajectories for 2-Player PEG (b) Cost Functions over Time

Figure 4.8: Evasion-weighted Objective, $\omega_G = 0.25, \omega_P = 0.75$. Capture at $t = 3.4$

weights (i.e., learn) between planning horizons based on the scenario.

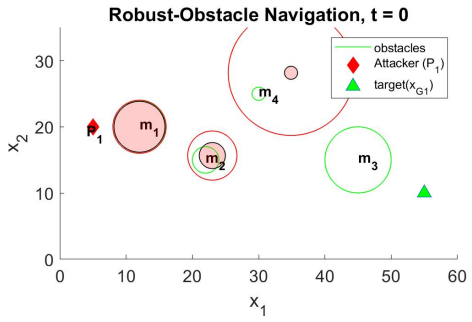
4.3.2 Learning via Robust Obstacles

By our definition of local information, only obstacles that player $\mathcal{P}_i$ can see should be represented by their nominal values $\hat{c}_m$ and $\hat{r}_m$. However, we can give the vehicles some information about future obstacles in the form of robust obstacles, and update the uncertainty bounds as it gets closer (e.g., as it gathers some information about the environment). In practice, this might look like the effective radii ${r^{[i]}}'_m$ of obstacles shrink as $\mathcal{P}_i$ approaches them, which is how we want it to conservatively conduct its path planning to maximize chances of a feasible strategy (X, U). The information about the robust-effective obstacles update each planning horizon, although the actual obstacles remain static. The effective radii ${r^{[i]}}'_m$, as defined by (4.9), provide a buffer around obstacles, the size of which is a function of the vehicle's proximity to the actual obstacle. An illustrative example in which the vehicle navigates around the effective obstacles which as shown in Figure 4.9, with

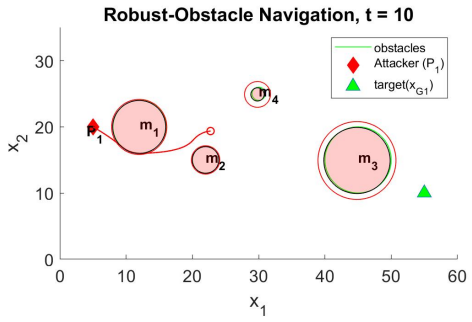
Table 4.2: Robust Navigation Parameters

Parameter	Value	Units	Description
t_0	0	s	start time
Δt	0.2	s	time step
T_h	10	s	horizon length
$x_{IC}^{[1]}$	$(5, 20, 0, 0)^T$	m	$\mathcal{P}_1$ initial state
x_G	$(55, 10, 0, 0)^T$	m	target point
v_{max}	2	m/s	max speed
a_{max}	10	m/s^2	max acceleration
$r_\delta^{[1]}$	30	m	detection range
ρ	0.7		confidence level
$\mathcal{O}_{(1)}$	$(12, 20, 4)^T$	m	obstacle data
$\mathcal{O}_{(2)}$	$(22, 15, 2)^T$	m	obstacle data
$\mathcal{O}_{(3)}$	$(45, 15, 5)^T$	m	obstacle data
$\mathcal{O}_{(4)}$	$(30, 25, 1)^T$	m	obstacle data

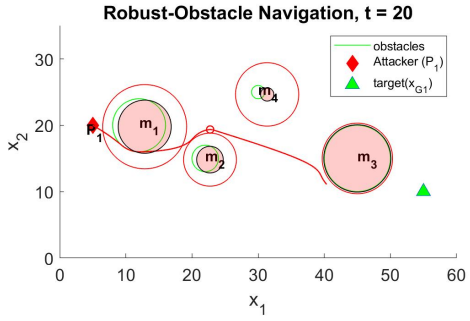
relevant parameters listed in Table 4.2. In Figure 4.9, the green circles represent the actual obstacles $\mathcal{O}_{(m)}$, while the red filled areas represent the detected nominal obstacles $\hat{\mathcal{O}}_{(m)}^{[1]}$, and the outer red rings represent the robust effective radius ${r^{[1]}}'_m$ around each nominal obstacle that the vehicle implements in its obstacle avoidance constraints. Note that in Sub-figure 4.9b, the furthest obstacle $\mathcal{O}_4$ is undetected, while the effective radii for near obstacles is close to their nominal radii (resolution is inversely proportional to proximity). Later, in Sub-figure 4.9c, the effective radius for $\mathcal{O}_1$ becomes larger since it is near the edge of the vehicle's detection range.



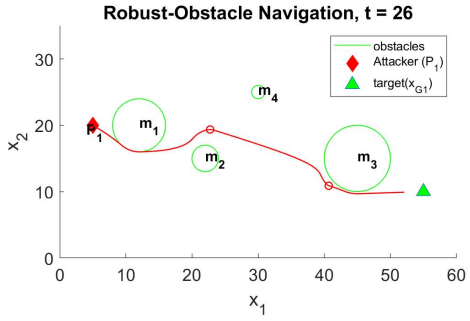
(a) Detected Obstacles in horizon $h = 1$



(b) Detected Obstacles in horizon $h = 2$



(c) Detected Obstacles in horizon $h = 3$



(d) Navigation through actual obstacles

Figure 4.9: Learning via Robust Obstacle Updates

Chapter 5

Game Expansion and Simulations for V&V

The previous chapters laid the foundation for equilibrium solutions to a two-player, pursuit-evasion game with robust obstacle avoidance and a stationary target. This provides a valuable benchmarking method of unmanned underwater vehicle (UUV) trajectory- and mission-planning. While the two-player model can be useful in safe trajectory validation, we can also modify this framework to model larger systems of UUVs for the validation of decoupled multi-vehicle mission plans (i.e., war-gaming) for current and future real-world applications.

In this chapter, we: 1) improve upon the current model by adding a minimum energy function to each player's cost functional, 2) introduce a system with multiple defenders, 3) explore the robustness of the current equilibrium model via Monte Carlo simulation, 4) compare to the non-linear programming time-optimal trajectory with deterministic obstacles and dynamics in single- and rolling-horizon scenarios, 5) discuss validation through equilibrium solutions.

5.1 Vehicle Platform

For the remaining simulations in this thesis, we utilize vehicle parameters from the Bluefin-21 Battlespace Preparation Autonomous Underwater Vehicle (BPAUV) [36], with the following relevant specifications. This platform is designed specifically

Table 5.1: Bluefin-21 UUV Specifications

Parameter	Value	Units	Description
T_{hmax}	18	h	endurance @ 3 kts
v_{max}	2	m/s	(approx. 4 kts) max speed
r_δ	150	m	sensor range (low resolution)

for U.S. Navy mine countermeasures (MCM) in shallow water, though has applications in anti-submarine warfare (ASW), which is most closely related to this study. The Bluefin-21 UUV serves as the basis for the Navy's Knifefish and Black Pearl UUVs currently being used and developed for research in both MCM and ASW.

5.2 Minimum-Energy Function

Upon introducing the vehicle detection concept in Chapter 4, it was discovered that the defender $\mathcal{P}_2$ consistently sought to find the origin at $(0, 0)$ when $\mathcal{P}_1$ was not within detection range $r_\delta^{[2]}$. Without the conditional pursuit sub-objective in the cost functional for $\mathcal{P}_2$'s optimization, the mixed complementarity problem (MCP) solution forced $\mathcal{P}_2$ to approach the origin. To mitigate the issue of unnecessary travel, we introduce a variant of the minimum-fuel objective commonly used in optimal control programs [35], that we define as the energy function:

$$f_E(x_k^{[i]}, u_k^{[i]}) = \frac{1}{2}\|u_k^{[i]}\|^2 \quad \forall \ k = 1, ..., N-1, \tag{5.1}$$

which is applied to each player's cost functional with an objective weighting parameter ω_E, which should be kept small (order of 10^{-3}). Combining this with the

sub-functions from attacker vehicle $\mathcal{P}_1$'s cost functional now a weighted sum of three sub-objectives, defined as:

$$J_h^{[1]}(s_h) \triangleq \frac{1}{2}\sum_{k=0}^{k_N}\left[\omega_G\|p_G - p_k^{[1]}\|^2 - \omega_P\gamma_2^{[1]}\|p_k^{[2]} - p_k^{[1]}\|^2 + \omega_E\|u_k^{[1]}\|^2\right], \tag{5.2}$$

while the objective function for the defender $\mathcal{P}_2$ is defined:

$$J_h^{[2]}(s_h) \triangleq \frac{1}{2}\sum_{k=0}^{k_N}\left[\gamma_1^{[2]}\|p_k^{[2]} - p_k^{[1]}\|^2 + \omega_E\|u_k^{[2]}\|^2\right], \tag{5.3}$$

The constraints remain the same, so this addition to the cost functional for each player only affects the MCP solution in the Lagrangian derivative with respect to the control variables as originally described in (3.5). The first two terms of the cost functional remain unchanged, while the last term is added in the rows pertaining to control variables $u_k^{[i]}$ in horizon h, so the gradient of the cost functional (3.5a) for player $\mathcal{P}_1$ now becomes:

$$\nabla_k J_h^{[1]} = \omega_G \begin{bmatrix} p_k^{[1]} - p_G \\ 0 \\ 0 \end{bmatrix} - \omega_P \begin{bmatrix} p_k^{[1]} - p_k^{[2]} \\ 0 \\ 0 \end{bmatrix} + \omega_E \begin{bmatrix} 0 \\ 0 \\ u_k^{[1]} \end{bmatrix} \quad \forall k \in K, \tag{5.4}$$

where the first term is the gradient with respect to $\mathcal{P}_1$'s state at each time step $x_k^{[1]}$ of the goal function, $\nabla_{x_k^{[1]}} f_G$; the second term is the gradient of the pursuit (evasion) function, $\nabla_{x_k^{[1]}}(-f_P)$; and the final term is the gradient with respect to control at each time step $u_k^{[1]}$ of the energy function, $\nabla_{u_k^{[1]}} f_E$.

Similarly for player $\mathcal{P}_2$, the pursuit function term of the cost functional does not change, but the minimal control sub-function is added. Now the gradient of the cost functional for player $\mathcal{P}_2$ is defined:

$$\nabla_k J_h^{[1]} = \begin{bmatrix} p_k^{[2]} - p_k^{[1]} \\ 0 \\ 0 \end{bmatrix} + \omega_E \begin{bmatrix} 0 \\ 0 \\ u_k^{[1]} \end{bmatrix} \quad \forall k \in K, \tag{5.5}$$

where the first term is the gradient with respect to $\mathcal{P}_2$'s state at each time step $x_k^{[2]}$ of the pursuit function, $\nabla_{x_k^{[2]}} f_P$; and the second term is the gradient with respect to control at each time step $u_k^{[2]}$ of the energy function, $\nabla_{u_k^{[2]}} f_E$.

5.3 Multiple-Defender, Single-Attacker Game

Building on the concepts presented in Section 3.2, we can extend the pursuit-evasion game to a system with more than one defender. So now let the number of attackers $|\mathcal{P}_\mathcal{A}| = 1$ and the number of defenders $|\mathcal{P}_\mathcal{D}| \geq 1$, meaning we now consider a $(1+|\mathcal{P}_\mathcal{D}|)$-player, non-cooperative game where the set of attackers $\mathcal{P}_\mathcal{A} = \{\mathcal{P}_1\}$ and the set of defenders $\mathcal{P}_\mathcal{D} = \{\mathcal{P}_2, ...\mathcal{P}_j, ..., \mathcal{P}_{N_\mathcal{I}}\}$. This allows us to investigate much larger, more complex problems, but we need to modify the attacker cost functional $J_h^{[1]}(s_{(h)})$, while also adjusting the rolling-horizon detection variables $\delta_1^{[j]}|_{j\in\mathcal{D}}$ for each defender to allow decoupled detection and control. In addition, since there are now teams of defenders (though still non-cooperative), a new friendly collision avoidance constraint (5.8f) needs to be added to the defenders' optimization problems.

The updated scenario involves an attacker UUV $\mathcal{P}_1$ whose mission it is to reach a stationary target position p_G as quickly as possible, but now a team of defender UUVs $\mathcal{P}_\mathcal{D}$ whose mission it is to intercept said attacker before reaching the target. Recall the pursuit function from (4.5) for any pair of players $\mathcal{P}_i$ and $\mathcal{P}_j$ at each time step k that is restated below:

$$f_P^{[i]} = \gamma_j^{[i]} \frac{1}{2} \|p_k^{[j]} - p_k^{[i]}\|^2, \tag{5.6}$$

which is again enforced only if vehicle detection is true, i.e. $\gamma_j^{[1]} = \gamma_1^{[j]} = 1$, where the detection equity assumption (4.2) still holds.

5.3.1 Attacker-Optimization Problem

With the addition of defenders, the evasion sub-objective in the attacker $\mathcal{P}_1$'s optimization problem needs to be normalized so that the total evasion weight remains equal to ω_P. To do so, we introduce the term $N_\mathcal{D}$, which represents the number of defenders $\mathcal{P}_{j\in\mathcal{D}}$ detected by the attacker $\mathcal{P}_1$. Thus, the second term of $\mathcal{P}_1$'s cost functional considers the weighted average of the evasion functions for $\mathcal{P}_1$ as it interacts with each of the defenders $\mathcal{P}_{j\in\mathcal{D}}$. This results in the modified cost

functional for $\mathcal{P}_1$ in each planning horizon now:

$$\min_{u_{(h)}^{[1]}} \; J_h^{[1]}(s) \triangleq \sum_{k=k_0}^{k_N} \left[\omega_G f_G(x_k^{[1]}) - \frac{\omega_P}{N_{\mathcal{D}}} \sum_{j=1}^{N_{\mathcal{D}}} f_P(x_k^{[1]}, x_k^{[j]}) + \omega_E f_E(u_k^{[1]}) \right] \tag{5.7a}$$

$$= \frac{1}{2} \sum_{k=k_0}^{k_N} \left[\omega_G \|p_G - p_k^{[1]}\|^2 - \frac{\omega_P}{N_{\mathcal{D}}} \sum_{j=1}^{N_{\mathcal{D}}} \gamma_j^{[1]} \|p_k^{[j]} - p_k^{[1]}\|^2 + \omega_E \|u_k^{[1]}\|^2 \right]$$

s.t.

$$h(s_h^{[1]}) \triangleq \mathbf{A}_h x_{(h)}^{[1]} + \mathbf{B}_h u_{(h)}^{[1]} - b_h^{[1]} = 0 \quad (\vec{\mu}_h^{[1]}) \tag{5.7b}$$

$$g_k^{vel}(x_k^{[1]}) \triangleq \frac{1}{2} \left(\left\| v_k^{[1]} \right\|^2 - {v_{max}^{[1]}}^2 \right) \leq 0 \quad \forall \; k \in K \quad (\sigma_k^{[1]}) \tag{5.7c}$$

$$g_k^{acc}(u_k^{[1]}) \triangleq \frac{1}{2} \left(\|u_k^{[1]}\|^2 - {a_{max}^{[1]}}^2 \right) \leq 0 \quad \forall \; k = 0, ..., N-1 \quad (\nu_k^{[1]}) \tag{5.7d}$$

$$g_{km}^{rob}(x_k^{[1]}) \triangleq \frac{1}{2} \left(r_m'^{[1]2} - \left\| p_k^{[1]} - \hat{c}_m^{[1]} \right\|^2 \right) \leq 0 \quad \forall \; k \in K, m \in M^{[1]} \quad (\lambda_{km}^{[1]}) \tag{5.7e}$$

where the state and control dynamics matrices $\mathbf{A}$ and $\mathbf{B}$ are as defined in (3.3), and the Greek symbols in $(\cdot)$ are the Lagrange multipliers for each constraint for $\mathcal{P}_1$. This is a weighted multi-objective quadratic program subject to quadratic constraints.

5.3.2 Defender-Optimization Problem

Now with multiple defenders, two items need to be addressed with respect to their optimization problems within the game scenario. First, we restate the definition for the detection indicator function

$$\gamma_j^{[i]} = \mathbf{1}\{\|p_{k_0}^{[i]} - p_{k_0}^{[j]}\| \leq r_\delta^{[i]}\}, \quad \forall i, j \in \{1, ..., |\mathcal{P}|\}, i \neq j, \tag{4.4}$$

which is now generalized across all players $\mathcal{P}_i$ and $\mathcal{P}_j, i \neq j$. Each player updates its detection variable for all other players, and we assume in this study that defenders do not share information about the attacker's state.

Assumption 5.1 (Decoupled Detection) *Defenders that detect an attacker do not share information about the attacker's state with the other defenders, i.e.,* $\delta_1^{[i]} = 1$ *does not imply* $\delta_1^{[j]} = 1 \ \ \forall j \neq i \in \mathcal{D}$.

Though underwater communication capabilities for cooperative teams of robots exist, the latency (time delay) of acoustic signals for data transfer present complications in real-time information sharing [1], necessitating the current research in decoupled/decentralized control [7].

Now, the defenders also need to ensure friendly collision avoidance with one-another, which requires the definition of a safety distance parameter d_{safe}, which can be interpreted as a circle around each defender [44]. Formally, this introduces an additional constraint per detected friendly player:

$$g_{jk}^{col[i]} = \gamma_j^{[i]} \frac{1}{2} \left[d_s^2 - \|p_k^{[i]} - p^{[j]}\|^2 \right] \leq 0 \quad \forall \ k \in K, \ \ i, j \in D, i \neq j, \tag{5.8f}$$

where both $\mathcal{P}_i$ and $\mathcal{P}_j$ are defender vehicles in this case, and $\gamma_j^{[i]}$ again is defined as in (4.4), so the constraint is only enforced if the players detect one-another.

This results in the updated optimization for each defender $\mathcal{P}_{j \in \mathcal{D}}$ as given below:

$$\min_{u_{(h)}^{[j]}} \; J_h^{[j]}(s_h) \triangleq \sum_{k=0}^{N} f_P(x_k^{[1]}, x_k^{[j]}) + \omega_E f_E(u_k^{[j]}) \tag{5.8a}$$

$$= \frac{1}{2} \sum_{k=k_0}^{k_N} \gamma_1^{[j]} \|p_k^{[j]} - p_k^{[1]}\|^2 + \omega_E \|u_k^{[j]}\|^2$$

s.t.

$$h(s_h^{[j]}) \triangleq \mathbf{A}_h x_{(h)}^{[j]} + \mathbf{B}_h u_{(h)}^{[j]} - b_h^{[j]} = 0 \quad (\vec{\mu_h}^{[j]}) \tag{5.8b}$$

$$g_k^{vel}(x_k^{[j]}) \triangleq \frac{1}{2}\left(\left\|v_k^{[j]}\right\|^2 - {v_{max}^{[j]}}^2\right) \leq 0 \quad \forall \; k \in K \quad (\sigma_k^{[j]}) \tag{5.8c}$$

$$g_k^{acc}(u_k^{[j]}) \triangleq \frac{1}{2}\left(\|u_k^{[j]}\|^2 - {a_{max}^{[j]}}^2\right) \leq 0 \quad \forall \; k = 0, ..., N-1 \quad (\nu_k^{[j]}) \tag{5.8d}$$

$$g_{km}^{rob}(x_k^{[j]}) \triangleq \frac{1}{2}\left(r_m'^{[j]2} - \left\|p_k^{[j]} - \hat{c}_m^{[j]}\right\|^2\right) \leq 0 \quad \forall \; k \in K, m \in M^{[j]} \quad (\lambda_{km}^{[j]}) \tag{5.8e}$$

$$g_{ik}^{col}(x_k^{[i]}, x_k^{[j]}) = \gamma_i^{[j]} \frac{1}{2}\left[d_s^2 - \|p_k^{[i]} - p_k^{[j]}\|^2\right] \leq 0 \quad \forall \; k \in K, \;\; i \neq j \in \mathcal{D} \;\; (\psi_{ik}^{[j]}) \tag{5.8f}$$

where each defender $\mathcal{P}_{j\in\mathcal{D}}$ pursues the attacker $\mathcal{P}_1$ while avoiding both obstacles and other defenders $\mathcal{P}_{i\in\mathcal{D}}$ within its detection range $r_\delta^{[j]}$. The cost functional (5.8a) includes the pursuit function if $\gamma_1^{[j]} = 1$ and minimum fuel sub-function as described by (5.3). The linear dynamics constraints, velocity and acceleration bounds (5.8b)-(5.8d) remain unchanged and generalize to all defenders. Both the obstacle avoidance (5.8e) and friendly collision avoidance constraints (5.8f) are conditional, in that (5.8e) is enforced if $m \in M^{[j]}$ and (5.8f) is enforced relative to $\mathcal{P}_i$ if $\gamma_i^{[j]} = 1$.

5.3.3 Equilibrium Solution to a Multi-Defender, Single-Attacker Game

The equilibrium trajectory solution to a two-defender, single-attacker pursuit-evasion game is depicted in Figure 5.1, where the attacker reaches its target at time $t_k = 20.8$ s). Figure 5.2 shows the evolution of each player's relevant sub-objectives over time.

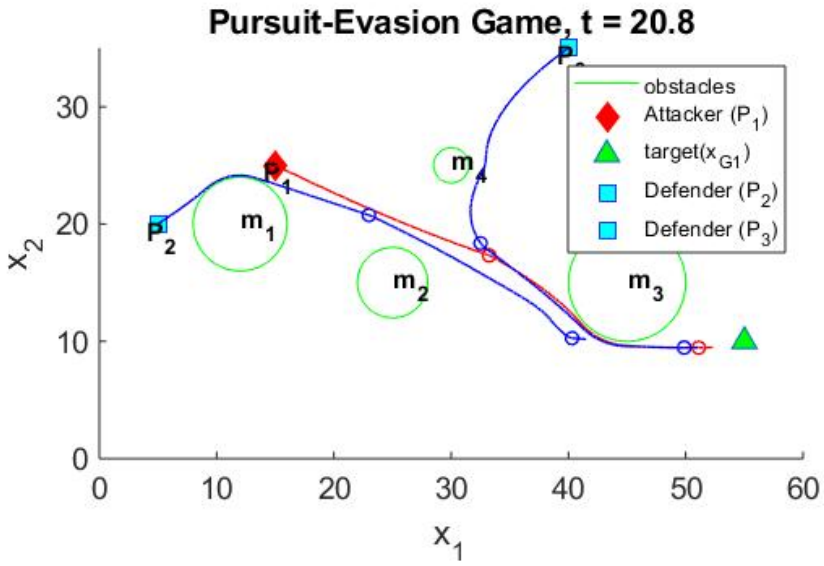


Figure 5.1: Pursuit-Evasion Game with two Defenders, one Attacker

The defenders' pursuit functions level off when the pursuers follow the same path as the attacker since all vehicles have the same max speed v_{max} in this scenario. Important to note in this formulation is that each individual vehicle operates in its own best interest, meaning that even if there are two defenders with the same objective to intercept an attacker, they operate independently.

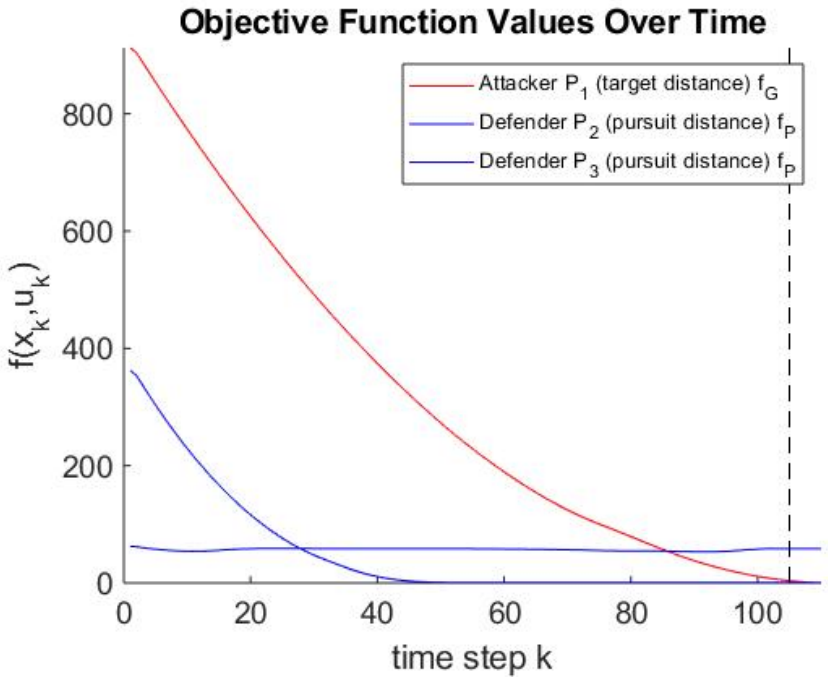


Figure 5.2: Objective Values for two Defender, one Attacker PEG

5.4 Verification of Effective Obstacles

The robust effective obstacle representation described in Chapter 4 is conservative, but still requires verification that the vehicles select feasible paths regardless of how the obstacles appear as nominal detected obstacles. This means that the effective obstacles must account for an infinite number of potential detected obstacles with nominal center $\hat{c}_m$ and radius $\hat{r}_m$ within the intervals for Δc_m and Δr_m that are expected based on actual obstacle realizations. To verify that the vehicles choose a safe path via the robust effective radii r'_m about detected centers $\hat{c}_m$, we implement a Monte Carlo simulation of a vehicle attempting to reach a stationary target p_G by navigating around 100 potential orientations of a single obstacle. We define success as the vehicle reaching the target without violating the obstacle avoidance constraint. Running the simulation for $r_m = 10, 20$, and 30 m with horizon lengths

$T_h = 60, 45$, and 30 seconds, we can observe what configurations cause errors in the robust obstacle approach. These errors are measured by the residual values generated by the GAMS PATH solver, which are on orders smaller than 10^{-6} when a safe trajectory is found, but reflect numbers larger than 1 when the resulting trajectory is infeasible. Table 5.2 describes the results.

Table 5.2: Robust Obstacle Verification

Obs. radius r_m (m)	Horizon T_h (sec)	Success rate* %
10	60	100%
	45	99.5%
	30	100%
20	60	99.7%
	45	96.4%
	30	94.5%
30	60	98.1%
	45	95.3%
	30	47.2%

*KKT point (i.e., viable solution)

It becomes clear that even using the robust obstacle approach, vehicles can encounter situations where they struggle to choose a direction around an obstacle, particularly when they are close to the surface of an obstacle and their target lies directly on the opposite side. When the vehicle's starting position, the obstacle center and target lie on the same line, the vehicle does not prioritize either direction and collides with the obstacle. This scenario seems to be most common when the planning horizon is short (say $T_h = 30$ sec) and a horizon terminal point x_{k_N} lands directly in front of the obstacle. In practice, this scenario is highly unlikely with random obstacles, and even incremental shifts from the line allow significant

improvement in performance (moving the obstacle down 0.5 meters in the case where $r_m = 30, T_h = 30$ in this example improves the success rate to 89%).

5.5 Numerical Verification of KKT Stationary Points

Another aspect of this model that is important to verify is under what conditions the model fails to generate a solution. There are potential situations in which the MCP is insufficient for the current formulation, specifically when the KKT optimality conditions do not hold. These scenarios have been seen to arise when the cost functional for the attacker $\mathcal{P}_1$ becomes non-convex, which we investigate via simulation. Using independent random initial conditions $x_{IC}^{[1]}$, $x_{IC}^{[2]}$ and targets p_G, we identify the effect of objective weights, specifically the goal weight ω_G on the success rate (i.e., relative frequency of generating a viable solution) of the game. By removing obstacles and simulating a two-player game over 1000 replications, while only varying ω_G, a trend emerges that sheds light on the cost functional $J^{[1]}$ for the attacker $\mathcal{P}_1$, which for two players is given by (5.2). This cost functional is a convex-concave combination which $\mathcal{P}_1$ seeks to minimize. Figure 5.3 illustrates the relationship between the attacker's cost functional $J^{[1]}$ and ω_G, where the box plots represent the quartiles for $J^{[1]}$ at each weight, and each red "+" is considered an outlier. The gray shaded region ($J^{[1]} < 0$) depicts when a KKT point is unlikely to be found.

Recall that the pursuit function weight $\omega_P = 1 - \omega_G$, so the negative values of $J^{[1]}$ when $\omega_G \leq 0.4$ is actually intuitive since the concave pursuit function is

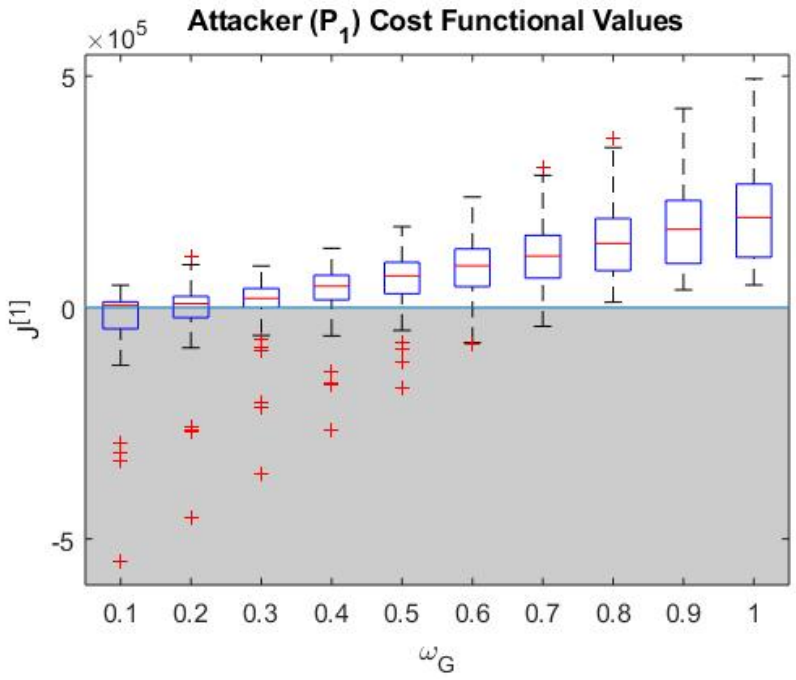


Figure 5.3: Verification of when KKT conditions hold

more heavily weighted in these scenarios, causing the overall cost functional to become concave. Thus, we identify a limitation in the formulation that the convex components of each player's cost functional (e.g., f_G, f_E) must outweigh the concave components (e.g. f_P for $\mathcal{P}_1$) such that $J^{[i]}(s) \geq 0 \ \forall i \in \mathcal{I}$ in order for the KKT conditions to hold. Further inspection of the attacker $\mathcal{P}_1$'s cost functional for the two-player pursuit-evasion game

$$J_h^{[1]}(s) = \frac{1}{2}\sum_{k=0}^{N}\left(\omega_G\|p_G - p_k^{[1]}\|^2 - \omega_P\gamma_2^{[1]}\|p_k^{[2]} - p_k^{[1]}\|^2 + \omega_E\|u_k^{[1]}\|^2\right)$$

can concretely describe how the pursuit function affects the reliability of KKT optimality. For instance, the Hessian matrix of second derivatives of $J_h^{[1]}$ for this game

with simple horizon where $N = 1 (k = 0, 1)$ is:

$$H = \begin{bmatrix} \omega_G - \omega_P \gamma_2^{[1]} & 0 & 0 & 0 & \omega_P \gamma_2^{[1]} & 0 \\ 0 & \omega_G - \omega_P \gamma_2^{[1]} & 0 & 0 & 0 & \omega_P \gamma_2^{[1]} \\ 0 & 0 & \omega_E & 0 & 0 & 0 \\ 0 & 0 & 0 & \omega_E & 0 & 0 \\ \omega_P \gamma_2^{[1]} & 0 & 0 & 0 & -\omega_P \gamma_2^{[1]} & 0 \\ 0 & \omega_P \gamma_2^{[1]} & 0 & 0 & 0 & -\omega_P \gamma_2^{[1]} \end{bmatrix}, \tag{5.9}$$

whose eigenvalues are

$$\lambda_H = \begin{bmatrix} \omega_E \\ \omega_E \\ \omega_G/2 - \omega_P \gamma_2^{[1]} - \frac{1}{2}\sqrt{\omega_G^2 + 4(\omega_P \gamma_2^{[1]})^2} \\ \omega_G/2 - \omega_P \gamma_2^{[1]} - \frac{1}{2}\sqrt{\omega_G^2 + 4(\omega_P \gamma_2^{[1]})^2} \\ \omega_G/2 - \omega_P \gamma_2^{[1]} + \frac{1}{2}\sqrt{\omega_G^2 + 4(\omega_P \gamma_2^{[1]})^2} \\ \omega_G/2 - \omega_P \gamma_2^{[1]} + \frac{1}{2}\sqrt{\omega_G^2 + 4(\omega_P \gamma_2^{[1]})^2} \end{bmatrix}. \tag{5.10}$$

It can be determined that no combination of ω_G and ω_P (except when $\omega_G = 1$, $\omega_P = 0$) exist such that all eigenvalues are non-negative, so we cannot claim the KKT conditions are sufficient for this cost functional. However, the simulated results imply that they are often necessary to find stationary equilibrium points, especially when $\omega_G \geq \omega_P$. Figure 5.4 shows the success rate (i.e., equilibrium point being found) for 2000 independent simulations varying $\omega_G \in [0.1, 1]$ and $\omega_E = 0.001, 0.005$.

Note that the numerically derived success rate for $\omega_G \geq 0.5$ is 100%. From these

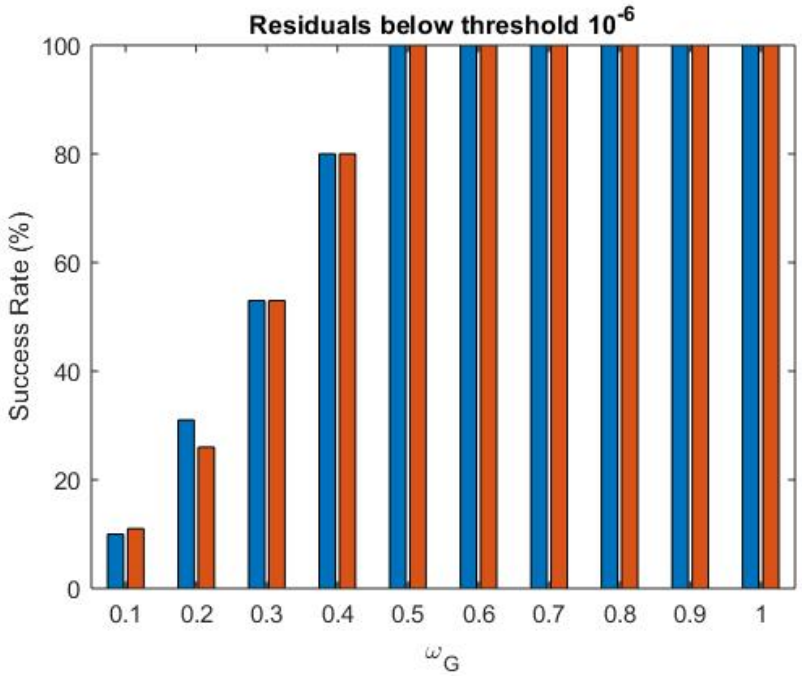


Figure 5.4: Success Rate of Two-player Game with no obstacles

simulations, we observe a positive correlation between KKT point being found (i.e. residuals $< 10^{-6}$) and a non-negative attacker cost functional $J^{[1]}$. However, there exist scenarios (5% of a 2000-simulation sample set) for which a stationary point can be found despite a negative cost functional. This relates to the fact that KKT conditions are not sufficient for this convex-concave cost functional.

5.6 Game Theoretic Validation for UUV Trajectories

The model presented in this thesis generates a solution to a pursuit-evasion game with a closed-loop information pattern with rolling-horizon foresight. Closed-loop information provides the "perfect-information reality" for game theoretic validation, in that each player $\mathcal{P}_{i\in\mathcal{I}}$ can simultaneously adjust its strategy for the current

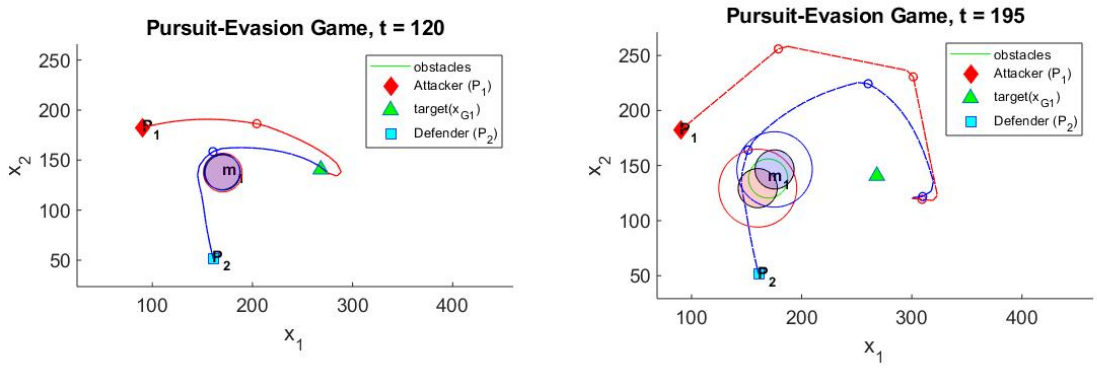
horizon based on the state of the rest of the system [19].

Observing the equilibrium solutions to these non-cooperative games where each player has access to closed-loop information about other players within detection range gives valuable insight towards UUV trajectory and mission planning, as it represents a sort of instantaneous online trajectory optimization for each player. Meanwhile, implementing the rolling-horizon robust obstacle avoidance for each vehicle creates a collision-free feasible game as long as the sparse obstacle assumption (4.4) holds.

5.6.1 Simulations of Equilibrium vs. Iterative Optimization

A common approach to modeling systems of non-cooperative unmanned vehicles is via open-loop iterative games, where the players make decisions based only on the initial conditions of other players, but with no information about other players' current decisions [19], [39]. For demonstration, we establish a test model as a two-player ($N_{\mathcal{I}} = 2$) Stackelberg-type game, where the attacker $\mathcal{P}_1$ selects a trajectory based on the target position p_G and the defender $\mathcal{P}_2$'s initial state $x_{k_0}^{[2]}$, while the defender $\mathcal{P}_2$ selects its paths with full knowledge of $\mathcal{P}_1$'s moves.

In the "asymmetric NLP" approach, the solution sequence for each planning horizon h is as follows: 1) Solve the attacker NLP (5.7) where the defender's state is fixed (i.e. $x_k^{[2]} = x_{k_0}^{[2]} \forall k \in K_h$), and save the attacker solution $(x_{(h)}^{[1]}, u_{(h)}^{[1]})$. 2) Solve the defender NLP (5.8) given the solution to $\mathcal{P}_1$'s NLP. This is repeated for each horizon until termination criteria (capture or goal) is met. In contrast, our



(a) MCP with perfect info (b) NLP with Asymmetric Info

Figure 5.5: Comparison of MCP vs. NLP Trajectories, $T_h = 60$ sec

MCP-derived equilibrium, which solves simultaneously, allows each player to act with perfect knowledge of other players' actions throughout the planning horizon, rather than only the initial conditions. The attacker $\mathcal{P}_1$'s trajectory becomes highly dependent on the length of the planning horizon, since its knowledge of the defender $\mathcal{P}_2$'s state updates only at the beginning of each horizon. Thus, as depicted in Figure 5.5, the attacker may take a far less direct path, increasing the likelihood of capture if the planning horizon T_h is relatively large.

To compare the MCP solution to the iterative optimization approach, we conduct a simulation with input random variates (RVs) $X_{IC}^{[1]}, X_{IC}^{[2]}, X_G, Y$ and Z, representing the initial positions $p_{IC}^{[i]}$ for each player, the target location p_G, the actual obstacle center c_m and radius r_m, as well as the detection error of obstacle center $\Delta c^{[i]}$ and radius $\Delta r^{[i]}$ in each replication, respectively. These RVs are summarized in Table 5.3.

The uniform probability distributions used in the simulation were selected to keep initial conditions arrayed to limit the number of trivial solutions where the

Table 5.3: Random Variates for MCP vs. NLP Comparison

Parameter	Random Variate	Description
$p_{IC}^{[1]}$	$X_{IC}^{[1]} \sim [\mathcal{U}(0,100), \mathcal{U}(0,200)]^T$	$\mathcal{P}_1$ initial position
$p_{IC}^{[2]}$	$X_{IC}^{[2]} \sim [\mathcal{U}(100,200), \mathcal{U}(0,200)]^T$	$\mathcal{P}_2$ initial position
p_G	$X_G \sim [\mathcal{U}(0,200), \mathcal{U}(0,300)]^T$	target location
$\mathcal{O}_1$	$Y \sim [\mathcal{U}(0,100), \mathcal{U}(0,200), \mathcal{U}(0,50)]^T$	actual obstacle attributes
$\Delta\mathcal{O}_1$	$Z \sim [\mathcal{U}(0,1), \mathcal{U}(0,2)]^T$	detected obstacle deviation

attacker can reach its target without influence from the defender or the obstacle. A visual representation of this simulation setup is shown in Figure 5.6. While varying

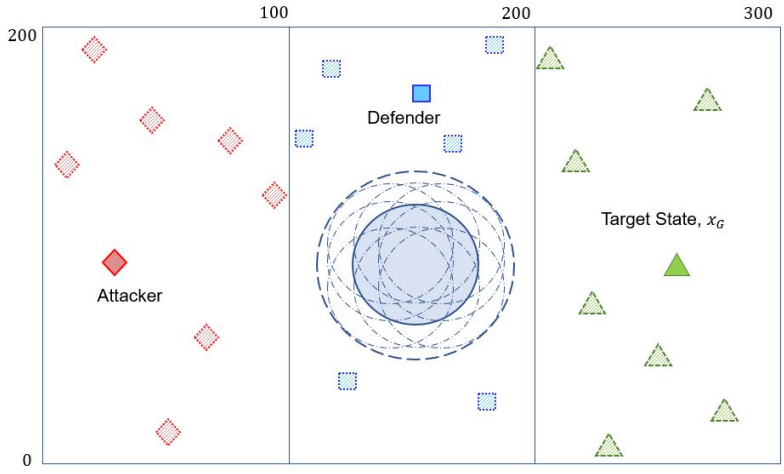


Figure 5.6: Workspace with randomly generated positions

the planning horizon length T_h, four performance measures are tracked for both solution types. These performance measures include the energy consumption for the attacker $\mathcal{P}_1$ over the entire path F_E, overall attacker path length F_D, capture rate ($\%Capture$) and clock time. We define $F_E = \sum_{h=1}^{H} \sum_{k \in K_h} f_E$ as the cumulative sum of all energy function values f_E (5.1) over all horizons h. The energy consumption seems to be slightly greater in the MCP solution than in nonlinear program approach, because the MCP solution makes incremental changes in direction

(accelerations) at each time step across the entire horizon, while the asymmetric NLP solution for the attacker only accelerates once per horizon based on $\mathcal{P}_2$'s initial conditions. In reality, energy is also expended as a function of propulsion to compete with external forces like drag and buoyancy, even at constant speed. Thus, we also investigate the expected value of the overall path length for the attacker $\mathcal{P}_1$ F_D, which is defined as the cumulative sum of the displacement function $d_h^{[1]}$ (4.1) over all horizons h.

Using common random numbers (CRNs), an equilibrium solution is found using the MCP method and performance measures stored for each simulation. The same CRNs are used to find a solution to the test model (asymmetric NLP sequence previously described). The results of this simulation are shown in Table 5.4 where

Table 5.4: Attacker Performance in Simulated Pursuit-Evasion Games

Solution Type	Horizon T_h (sec)	% Capture	$\mathbf{E}(F_E)$	$\sigma(F_E)$	$\mathbf{E}(F_D)$	$\sigma(F_D)$
MCP	30	28.6%	0.822	0.798	1694	284
	45	40.3%	0.917	0.529	1769	194
	60	51.5%	0.690	0.428	1798	347
	75	53.1%	0.869	0.552	1835	130
NLP	30	20.1%	0.078	0.011	1269	151
	45	48.4%	0.192	0.169	2073	69
	60	70.9%	0.264	0.018	2731	100
	75	77.9%	0.274	0.023	3042	103

T_h is the finite designated horizon length for each rolling horizon h, $\mathbf{E}(F_E)$ is the total energy consumption (represented in acceleration units - m/s^2) and $\mathbf{E}(F_D)$ is the overall path length. The $\mathbf{E}(\cdot)$ and $\sigma(\cdot)$ represent the mean and standard deviation of the simulation sample performance measures, respectively. These results are also

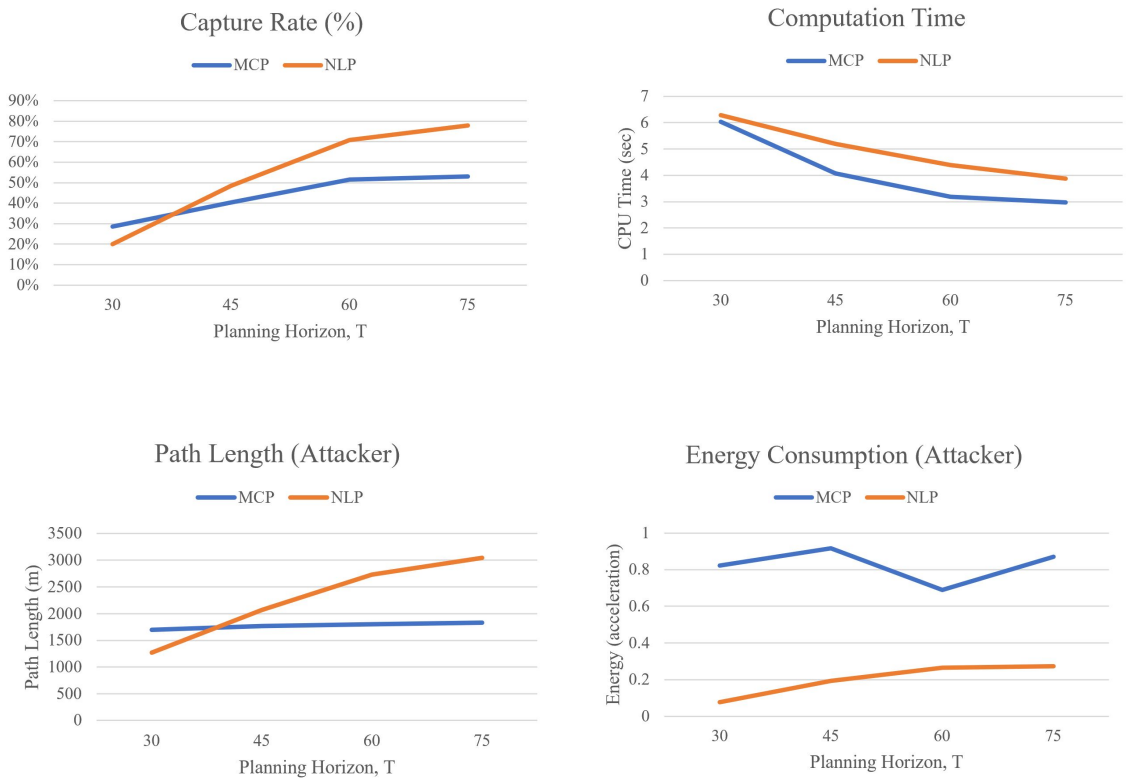


Figure 5.7: Performance Measures vs. Planning Horizon Length, T_h

illustrated graphically in Figure 5.7, where it can be seen that the MCP solution tends to provide a lower bound in both capture rate and path length. The simulation also sheds light on the approximate planning horizon range ($30 \leq T_h \leq 40$ seconds) where the gap in performance between the two models is minimal.

Chapter 6

Conclusions

6.1 Summary of Work

This thesis presents a formulation and an algorithm for modeling single- and multi-agent game theory-based trajectory planning with rolling-horizon foresight and robust obstacles in two-dimensional space. Chapter 2 discusses a nonlinear trajectory optimization program is presented, along with fundamental game theoretic concepts. This trajectory optimization is solved via a mixed complementarity problem (MCP) formulation using the Karush-Kuhn-Tucker (KKT) optimality conditions in Chapter 3, followed by the introduction of a pursuit-evasion game (PEG) scenario for a single attacker, single defender and stationary target. Rolling-horizon foresight and robust obstacles are incorporated into the model in Chapter 4, both of which improve model performance in determining feasible solutions. In Chapter 5, several improvements are made to the algorithm and simulation results are presented.

6.2 Suggestions for Future Work

There is enormous potential yet to be investigated in the use of robust games for spatio-temporal system validation. The mixed complementarity problem (MCP)

is a powerful tool in the analysis of mechanical systems, and provides valuable insight if used for benchmarking purposes. This study just skims the surface of robust, game-driven trajectory planning, and can potentially be extended in the following directions:

6.2.1 Dealing with Non-convexity

The non-convex obstacle avoidance constraints (2.1f) imposed on each player in this thesis can result in a scenario that is unsolvable via MCP. There exist several iterative schemes that convexify these constraints for a single vehicle trajectory optimization problem via lossless convexification known generally as Sequential Convex Programming [28], [12], [44]. While these might be extended to the larger system of vehicles presented in this thesis, the obstacle constraints could also be addressed via linear norms (e.g., L_∞ and L_1 norms), though these require the introduction of additional variables in as penalty sub-functions in the objective.

The L_∞ norm seems to be the most straight-forward approach to dealing with the concave quadratic constraint associated with obstacle avoidance (2.1f). The L_∞ norm can be used to replace the non-convex obstacle avoidance constraint with

$$g_{km}^{obs}(x_k) = r_m - z_m \leq 0 \quad \forall \; k \in (1,..,K), \;\; m \in (1,...,M) \quad (\lambda_{km}) \tag{6.1a}$$

$$y_{km}^{+} + y_{km}^{-} \leq z_m \quad \forall \; k \in (1,..,K), \;\; j \in (1,...,M) \quad (\alpha_{km}) \tag{6.1b}$$

$$y_{km}^{+} - y_{km}^{-} = p_k - c_m \quad \forall \; k \in (1,..,K), \;\; m \in (1,...,M) \quad (\beta_{km}) \tag{6.1c}$$

where r_m is the obstacle radius, and y_{km} is the difference between the vehicle position p_k and obstacle center c_m at each time step k, and z_m is the maximum value of all the difference variables y_{km} for each obstacle m. This also requires z_m be added to the cost functional $J(s)$ as a penalty function.

In any case, the KKT conditions should be further investigated in future work by determining if and when the Mangasarian-Fromovitz constraint qualification (MFCQ) holds for any given scenario.

6.2.2 Inevitable Collision States

Though solving our system iteratively over rolling horizons improves computational efficiency and overall performance, there are inherent risks that this method could possibly select a trajectory with an unsafe terminal state in horizon h that leads to a collision in horizon $h+1$, called an inevitable collision state [15], [5], [21]. In practice, this situation arises when a vehicle travels toward its target, but provided a finite horizon, does not consider the obstacle directly in front of it when it reaches its terminal state x_{k_N}. Thus on the subsequent horizon, its velocity causes it to collide with an obstacle in the next horizon, resulting in an infeasible solution. While the algorithm presented in this thesis does not directly address this issue and guarantee avoidance from inevitable collision, the adaptive, robust obstacles presented in Chapter 4 mitigate this risk almost entirely due to how the effective obstacles update (e.g., shrink or shift) between horizons.

6.2.3 Software Interface

The MATLAB-GAMS software interface chosen to implement this model carries with it several limitations, but provide quite a bit of flexibility once overcome. First of all, neither are open source platforms. For academic research, this is no serious problem since MATLAB is provided through the university and the GAMS license required to run the PATH solver can be requested directly from the GAMS sales team. Second, the GAMS sets and variable declaration requirements add a great deal of complexity and require an entire MATLAB script to prepare. However, once this step is complete, the solve speed and accuracy are exceptional. Continued development on the interface between these two packages would be beneficial to continued research on spatio-temporal games, as would a game theory-focused general validation package.

6.2.4 Vehicle-based Rigid Body Dynamics

To improve our formulation further, we can implement vehicle-based dynamics by shifting from the Cartesian "earth-fixed" workspace perspective to a vehicle-centric polar coordinate perspective. The vehicle state tuple would become $x = (p_1, p_2, \theta, v)$, where θ denotes the vehicle orientation and v represents vehicle speed,

while the control tuple is now $u = (u_v, u_\theta)$. Thus, the dynamics are defined

$$\dot{x} = \begin{bmatrix} \dot{p_1} \\ \dot{p_2} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ u^\theta \\ u^v \end{bmatrix}. \tag{6.2}$$

This approach introduces significant non-convexity due to the trigonometric functions, but is a natural transition to relating this research to optimal control with greater degrees of freedom.

Bibliography

[1] Yannick Allard and Elisa Shahbazian. Unmanned Underwater Vehicle (UUV) Information Study. page 78, 2014.

[2] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming Theory and Algorithms*. 2009.

[3] Stephen C. Billups and Steven P. Dirkse. A Comparison of Algorithms for Large Scale Mixed Complementarity Problem Solvers. *Computational Optimization and*, 23:1–23, 1995.

[4] Stephen P Boyd. *Convex Optimization Theory*, volume 25. 2010.

[5] Nicholas Chan, James Kuffner, and Matthew Zucker. Improved Motion Planning Speed and Safety using Regions of Inevitable Collision. *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, pages 103–114, 2008.

[6] GAMS Development Corporation. General Algebraic Modeling System (GAMS) Release 24.2.1. Washington, DC, USA, 2013.

[7] Lockheed Martin Corporation. Unmanned Underwater Vehicle Collaborative Missions A Decentralized Approach to Operating UUV Teams. (September), 2017.

[8] Mark Cutler, Jonathan P How Decoupled, Automation Icra May, Washington State, Yufan Chen, Mark Cutler, and Jonathan P How. Decoupled Multiagent Path Planning via Incremental Sequential Convex Programming. In *Proc. of the Intl. Conf. on Robot. and Autom.*, pages 5954–5961, 2016.

[9] Department of Defense. Unmanned Systems Integrated Roadmap 2017-2042. Technical Report August, 2016.

[10] Department of the Navy. The Navy Unmanned Undersea Vehicle (UUV) Master Plan. (November):127, 2004.

[11] Johannes Dorfner. GAMS and how to use it from MATLAB. `https://gams-matlab.readthedocs.io/en/latest/`, 2018.

[12] Daniel Dueri, Yuanqi Mao, Zohaib Mian, and Jerry Ding. Trajectory Optimization with Inter-sample Obstacle Avoidance via Successive Convexification. (Cdc):1150–1156, 2017.

[13] Bartomiej Jzef Dzieńkowski, Christopher Strode, and Urszula Markowska-Kaczmar. Employing Game Theory and Computational Intelligence to Find the Optimal Strategy of an Autonomous Underwater Vehicle against a Submarine. *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*, 8:31–40, 2016.

[14] Antonio Evangelio. Combined Joint Operations from the Sea Centre of Excellence. (July), 2012.

[15] Thierry Fraichard and Hajime Asama. Inevitable Collision States - A Step Towards Safer Robots ? *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems October*, 18(10):2–9, 2008.

[16] Giacomo Bonanno. *Game Theory*. 2nd edition, 2018.

[17] L. Giovanini, J. Balderud, and R. Katebi. Autonomous and Decentralized Mission Planning for Clusters of UUVs. *International Journal of Control*, 80(7):1169–1179, 2007.

[18] Haomiao Huang, Wei Zhang, Jerry Ding, Duan M. Stipanović, and Claire J. Tomlin. Guaranteed decentralized pursuit-evasion in the plane with multiple pursuers. *Proceedings of the IEEE Conference on Decision and Control*, pages 4835–4840, 2011.

[19] Sertac Karaman. *Principles of Autonomy and Decision Making; Lecture 25: Differential Games*. 2010.

[20] Matthew Kelly. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. *SIAM Review*, 59(4):849–904, 2017.

[21] Steven Lavalle. Planning Algorithms. 1999.

[22] Mian Li, Steven A. Gabriel, Yohan Shim, and Shapour Azarm. Interval Uncertainty-Based Robust Optimization for Convex and Non-Convex Quadratic Programs with Applications in Network Infrastructure Planning. *Networks and Spatial Economics*, 11(1):159–191, 2011.

[23] Tianchen Liu, Kevin Quigley, Shapour Azarm, and Steven Gabriel. Towards V&V of Autonomous Systems via Robust Optimization and Games. Lockheed Martin TIM Presentations, 2018.

[24] Yimeng Lu. A Game-theoretical Approach for Distributed Cooperative Control of Autonomous Underwater Vehicles. 2018.

[25] Yuanqi Mao, Daniel Dueri, Michael Szmuk, and Behet Açkmeşe. Successive Convexification of Non-Convex Optimal Control Problems with State Constraints. *IFAC-PapersOnLine*, 50(1):4063–4069, 2017.

[26] Yuanqi Mao, Michael Szmuk, and Behcet Acikmese. Successive Convexification of Non-Convex Optimal Control Problems and Its Convergence Properties. In *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 3636–3641, 2016.

[27] Yuanqi Mao, Michael Szmuk, and Behcet Açikmeşe. A Tutorial on Real-time Convex Optimization Based Guidance and Control for Aerospace Applications. *Proceedings of the American Control Conference*, 2018-June:2410–2416, 2018.

[28] Yuanqi Mao, Michael Szmuk, and Behcet Acikmese. Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems. pages 1–35, 2018.

[29] MATLAB. *Version 9.5 (R2018b)*. The MathWorks Inc., Natick, Massachusetts, 2018.

[30] Jerome Milgram, Christopher Von Alt, and Timothy Prestero. Verification of a Six-Degree of Freedom Simulation Model for the REMUS100 AUV. 2001.

[31] Stefan Mitsch, Khalil Ghorbal, David Vogelbacher, and Andr Platzer. Formal Verification of Obstacle Avoidance and Navigation of Ground Robots. *International Journal of Robotics Research*, 36(12):1312–1340, 2017.

[32] Rushen B. Patel and Paul J. Goulart. Trajectory Generation for Aircraft Avoidance Maneuvers Using Online Optimization. *Journal of Guidance, Control, and Dynamics*, 34(1):218–230, 2010.

[33] Warren B. Powell. Perspectives of Approximate Dynamic Programming. *Annals of Operations Research*, 241(1-2):319–356, 2016.

[34] Angel Rabasa, Peter Chalk, Kim Cragin, Sara A. Daly, Heather S. Gregg, Theodore W. Karasik, Kevin A. OBrien, and William Rosenau. A Survey of Missions for Unmanned Undersea Vehicles. Technical report, Santa Monica, CA, 2010.

[35] Arthur Richards, Tom Schouwenaars, Jonathan P. How, and Eric Feron. Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, 2008.

[36] Bluefin Robotics. Bluefin-21 BPAUV Product Sheet, 2010.

[37] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How. Mixed Integer Programming for Multi-Vehicle Path Planning. (3):2603–2608, 2001.

[38] Sumeet Singh, Anirudha Majumdar, Jean-Jacques Slotine, and Marco Pavone. Robust Online Motion Planning via Contraction Theory and Convex Optimization. pages 5883–5890, 2017.

[39] Ryo Takei, Haomiao Huang, Jerry Ding, and Claire J. Tomlin. Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation. In *2012 IEEE International Conference on Robotics and Automation*, pages 323–329. IEEE, 5 2012.

[40] Ryo Takei, Richard Tsai, Zhengyuan Zhoux, and Yanina Landa. An Efficient Algorithm for a Visibility-Based Surveillance-Evasion Game. *Communications in Mathematical Sciences*, 12(7):1303–1327, 2014.

[41] Vladimir Turetsky and Valery Y. Glizer. Defender-Attacker-Target Game: Open-Loop Solution. 2018.

[42] Alexander Von Moll, David Casbeer, Eloy Garcia, Dejan Milutinović, and Meir Pachter. The Multi-pursuer Single-Evader Game. *Journal of Intelligent & Robotic Systems*, (January), 2019.

[43] Eric W. Weisstein. "Pythagorean Triple" Mathworld–A Wolfram Web Resource. `http://mathworld.wolfram.com/PythagoreanTriple.html`, 2017.

[44] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-Based Collision Avoidance. pages 1–27, 2017.

[45] Zhe Zhang, Jianxun Li, and Jun Wang. Sequential Convex Programming for Nonlinear Optimal Control Problems in UAV Path Planning. *Aerospace Science and Technology*, 76:280–290, 2018.